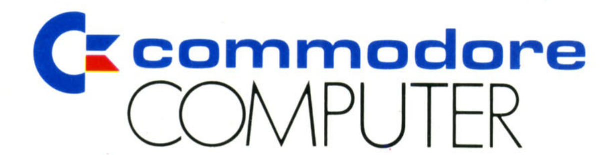
Commodore 64 MicroComputer

HANDBUCH



COMMODORE 64

Bedienungshandbuch



INHALTSVERZEICHNIS

Anhang		**Seite**

EINLEITUNG:

Wir möchten Sie hiermit beglückwünschen, daß Sie nun Besitzer eines COMMODORE 64 sind, einem der besten Mikrocomputer der Welt. Commodore ist bekannt als die Firma, die "Freundliche Computer" herstellt; die von Commodore herausgegebenen Handbücher zeichnen sich dadurch aus, daß sie leicht zu lesen, leicht zu verstehen und demzufolge leicht anzuwenden sind. Das COMMODORE 64 HANDBUCH vermittelt alle Informationen, um Ihre Anlage sachgemäß aufzubauen und mit dem Computer vertraut zu werden. Weiterhin lernen Sie auf einfache, abwechslungsreiche Weise, eigene Programme zu schreiben. Für alle Besitzer eines COMMODORE 64, die sich nicht gleich ins Programmieren stürzen wollen, haben wir diejenigen Informationen gleich zu Anfang zusammengefaßt, die Sie brauchen, um fertige Programme zu benutzen. Wenn Sie wollen können Sie also gleich anfangen; Sie brauchen nicht vorher das ganze Buch durchzulesen.
Nun wollen wir kurz beschreiben, was Ihr COMMODORE 64 alles kann. Wenn es um Grafik geht - Sie besitzen einen der höchstentwickelten Mikrocomputer der Welt. Die Konstrukteure des COMMODORE 64 haben die Methode der SPRITES erfunden, mit der Sie dreifarbige Figuren erstellen können, wie Sie sie aus professionellen Videospielen kennen. Nicht nur, daß Sie bis zu 8 verschiedene SPRITES erstellen können, Sie können die selbst entworfenen Figuren auf einfache Weise auf dem Bildschirm bewegen und können ein SPRITE vor dem anderen vorbeiziehen lassen. Sie können Ihren COMMODORE 64 auch damit beauftragen, das Zusammenstoßen zweier SPRITES zu registrieren, um (z.B. im Rahmen eines Bildschirmspiels) bestimmte vorprogrammierte Aktionen damit zu verknüpfen. So können Sie z.B. beim Zusammenstoß zweier "Raumschiffe" eine Explosion auslösen.
Weiterhin hat der COMMODORE 64 eingebaute Musik- und Toneffekte, die aus ihm einen vollwertigen Synthesizer machen. Dieser Teil Ihres Computers bietet Ihnen nämlich 3 voneinander unabhängige "Stimmen", von denen jede volle 8 Oktaven Umfang hat. Außerdem stehen Ihnen 4 verschiedene Wellenformen (Rechteck, Sägezahn, Dreieck, Rauschen),

ein programmierbarer Hüllkurven-Generator, programmierbare Hoch-, Tief- und Bandpaßfilter, sowie Lautstärke- und "Resonanz"-Regler zur Verfügung. Wenn Sie Ihre Musik in professioneller Qualität wiedergeben wollen, so können Sie Ihren COMMODORE 64 an jede Hi-Fi-Anlage anschließen.
Wenn wir schon beim Anschließen des COMMODORE 64 an andere Geräte sind... Ihr Computer kann durch Hinzufügen von Peripheriegeräten zum System erweitert werden. Sie können einen Kassettenrekorder oder bis zu 5 Diskettenlaufwerke vom Typ VC 1541 anschließen, um Programme zu speichern oder in den Computer zu laden (wenn Sie schon ein Laufwerk vom Typ VC 1540 besitzen, so können Sie es durch Ihren Händler "modernisieren" lassen). Sie können mit dem VC-Drucker Ihre Programme, Briefe usw. ausdrucken und - wenn sie an CP/M-Software interessiert sind - der COMMODORE 64 kann mit einer CP/M-Karte ausgerüstet werden, die einen Z 80-Mikroprozessor enthält.
Ebenso wichtig wie die zur Verfügung stehenden Geräte, die sogenannte Hardware, ist dieses Handbuch, das zum Verständnis Ihres Computers beitragen soll. Es enthält nicht alles, was man über Computer und über das Programmieren wissen sollte, aber es wird Sie in die Lage versetzen, an Hand weiterführender Literatur tiefer in die Materie einzudringen. Commodore möchte, das Sie Spaß an Ihrem COMMODORE 64 haben und wir haben auch versucht alles zu tun, um Ihnen dabei zu helfen. Aber denken Sie daran: Programmieren ist keine Fähigkeit, die man an einem Tag erlernt; fassen Sie sich in Geduld und gehen Sie Ihr Handbuch Punkt für Punkt sorgfältig durch. Noch eins - nehmen Sie sich ein paar Minuten Zeit, füllen Sie die Garantiekarte aus und schicken Sie sie ab, damit Ihr Gerät ordnungsgemäß registriert werden kann.

Und nun: Viel Spaß mit Ihrem COMMODORE 64.

Ein Hinweis: Während dieses Handbuch produziert wird, werden weltweit Programme für den COMMODORE 64 entwickelt. Fragen Sie ab und zu mal bei Ihrem Händler nach, was es Neues gibt.

KAPITEL 1

ANSCHLUSS UND INBETRIEBNAHME

DER ANSCHLUSS DES COMMODORE 64

Im folgenden wird beschrieben, wie Sie den COMMODORE 64 an ein Fernsehgerät, einen Verstärker oder einen Monitor anschließen können. Überprüfen Sie zuerst den Inhalt der Verpackung, in der der COMMODORE 64 geliefert wurde. Außer dieser Betriebsanleitung sollte die Packung folgendes enthalten:

1. COMMODORE 64
2. Netzgerät (Kästchen mit Netz- und Gerätekabel)
3. Antennenkabel

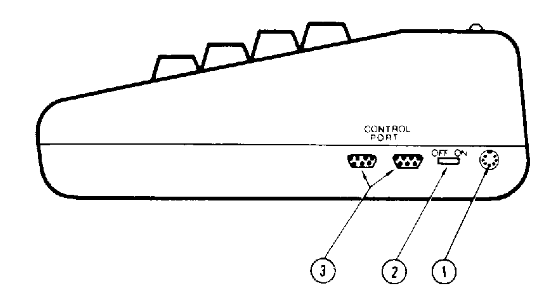
Vermissen Sie einen der aufgeführten Gegenstände, so fordern Sie bei Ihrem COMMODORE-Fachhändler Ersatz an.
Wir wollen uns nun als erstes anschauen, wo die verschiedenen Anschlüsse des Rechners sind und welche Funktion sie haben.

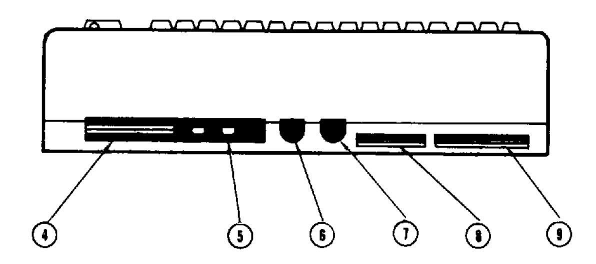
ANSCHLÜSSE AN DER RECHTEN SEITE DES GEHÄUSES

1. **Spannungsversorgung**. Hier wird der Gerätestecker des Kabels angeschlossen, das den Rechner mit dem Netzgerät verbindet.
2. **Ein/Aus-Schalter**
3. **Steuereingänge für Spiele.** An jeden der beiden Eingänge kann ein Joystick, ein Paddle oder ein Lichtgriffel angeschlossen werden.

ANSCHLÜSSE AUF DER RÜCKSEITE DES GEHÄUSES

4. **MODUL-STECKPLATZ.** Rechteckige Aussparung zum Einstecken der Programm- oder Spielmodule.
5. **TV-Anschluß.** Anschlußmöglichkeit an ein Fernsehgerät über den Antennenanschluß (75 Ohm). Das Signal enthält sowohl die Bild- wie auch die Toninformation.
6. **Audio & Video-Ausgang.** Hier kann das isolierte Tonsignal abgenommen und einer HI-FI-Anlage zugeführt werden. Außerdem steht ein Videosignal zur Verfügung, mit dem ein Monitor angesteuert werden kann.





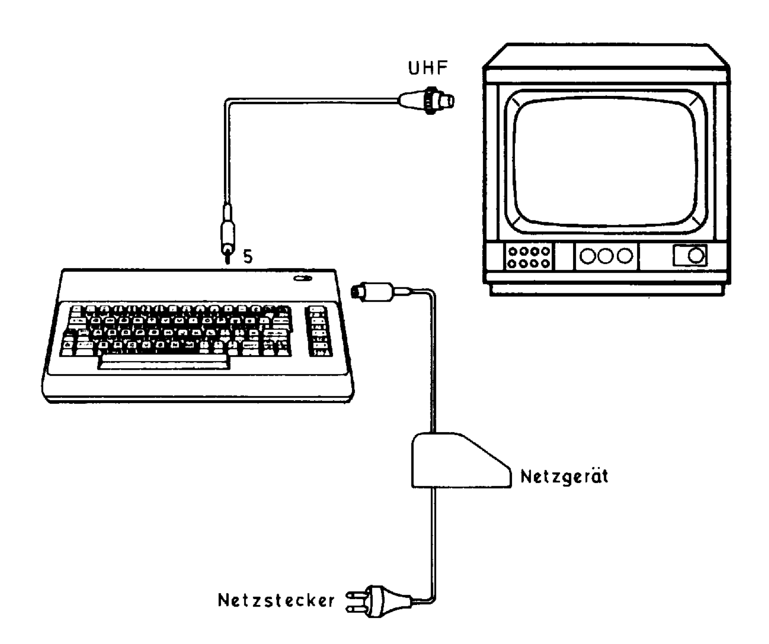
7. **Serieller Ausgang.** Zum Anschluß eines Druckers oder eines Disketten-Einzellaufwerks.
8. **Recorder-Anschluß.** Anschlußmöglichkeit für die Datasette zur Speicherung von Programmen oder Daten.
9. **User Port.** Frei programmierbarer Ein/Ausgang und Anschlußmöglichkeit für Steckmodule wie z. B.: RS-232-Schnittstelle.

ANSCHLUSS UND INBETRIEBNAHME

DER ANSCHLUSS AN DAS FERNSEHGERÄT

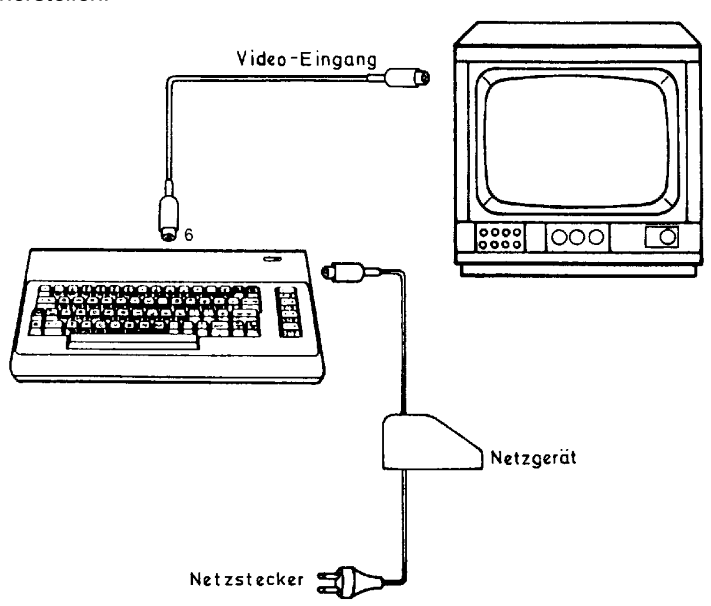
Das Schema für den Anschluß an das Fernsehgerät können Sie der Abbildung unten entnehmen.

1. **Verwendung des UHF-Antenneneingangs**
   Verbinden Sie mit Hilfe des beiliegenden Fernsehkabels den TV-Anschluß (Nr. 5) auf der Rückseite des COMMODORE 64 mit dem UHF-Antenneneingang Ihres Fernsehgerätes. Antenne vorher entfernen. Stellen Sie Ihr Gerät auf Kanal 36 ein.



2. Video-Ausgang

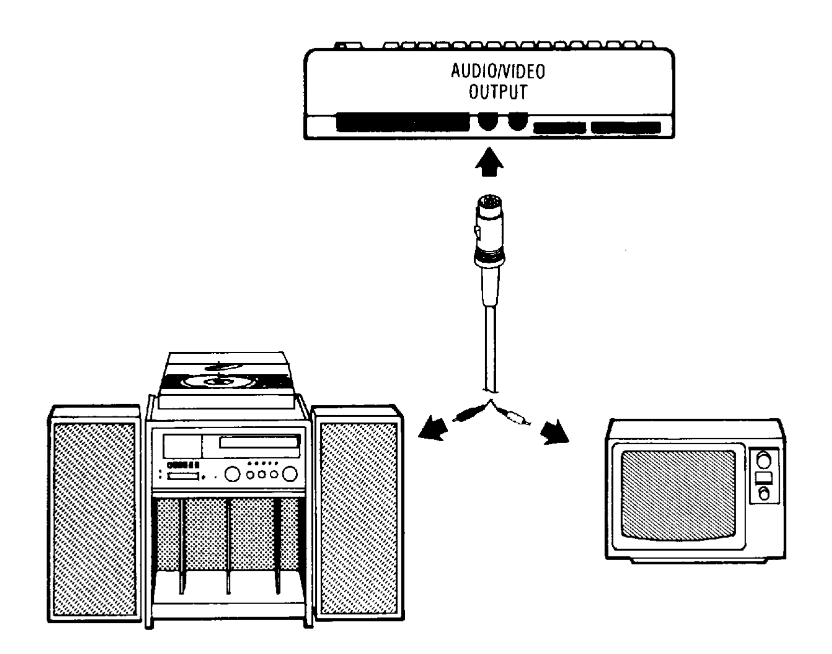
Das beste Bild erzielen Sie über einen Farb-Monitor. Verwenden Sie hierzu ein Videokabel, das an der einen Seite einen 5-poligen Diodenstecker (DIN 41524) für die Buchse 6 Ihres COMMODORE 64 hat, auf der anderen Seite einen Videostecker (DIN 45322) für Ihren Monitor. Falls Ihr Fernseher eine Video-Buchse hat, können Sie ihn ebenfalls als Monitor verwenden, Sie müssen nur dafür sorgen, daß der Video-Anschluß Ihres Fernsehers als Eingang geschaltet wird. Dies geschieht bei Verwendung eines Video-Recorders in der Regel durch eine Hilfsspannung von 12 V, die an Pin 1 der Video-Buchse des Fernsehers gelegt wird.
Ihr COMMODORE 64 gibt eine solche Hilfsspannung nicht ab. Lassen Sie sich daher von Ihrem Fernseh-Fachmann eine geeignete Umschaltung einbauen. Bei manchen Fernsehern genügt dazu schon eine Drahtbrücke im Video-Stecker, da bei manchen Fernsehern an Pin 5 der Video-Buchse die benötigte Hilfsspannung von 12 V bereitgestellt wird. Benutzen Sie daher diesen Eingang mit äußerster Vorsicht! Die Hilfsspannung darf auf keinen Fall an die Ausgänge Ihres COMMODORE 64 gelangen. Der Rechner würde dadurch sofort zerstört. Lassen Sie die Verbindung nur durch einen Fachmann herstellen!



Zum Empfang normaler Fernseh-Sendungen müssen Sie das Video-Kabel in der Regel entfernen.

3. **Netzanschluß**
   Verbinden Sie Ihr Netzgerät mit dem Commodore 64 (Buchse 1) und mit einem 220 V / 50 Hz-Netzanschluß.

4. **Weitere Anschlüsse**
   Da der COMMODORE 64 das Tonsignal in Hi-Fi-Qualität liefert, bietet es sich an, dieses Signal durch einen hochwertigen Verstärker zu verarbeiten. Das Tonsignal wird ebenfalls an Buchse 6 abgenommen (Belegung) der Steckerkontakte siehe Anhang I).



Wie Sie vielleicht wissen, können Sie als Peripherie z. B. noch das Diskettenlaufwerk ("Floppy Disk") VC 1541 oder den Drucker VC 1525 anschließen. Ein solches komplettes System kann etwa wie folgt aussehen.



INBETRIEBNAHME

1. Schalten Sie den Rechner mit Hilfe des Kippschalters auf der rechten Seite ein.
2. Nach einigen Sekunden muß der Bildschirm folgendes zeigen.

```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
■
```

Der blinkende Cursor zeigt, daß der COMMODORE 64 auf Ihre Eingabe wartet.

3. Wenn die Senderfeineinstellung Ihres Fernsehgeräts von Hand erfolgt, so müssen Sie selbst eine Einstellung suchen, die ein optimal scharfes Bild ergibt. Wenn Ihr Gerät eine Automatik besitzt, nimmt es Ihnen diese Arbeit ab.
4. Die Einstellung der Farben und der Helligkeit des Bildschirms ist im nächsten Abschnitt beschrieben. Ein erster Anhaltspunkt für die richtige Einstellung Ihres Fernsehgerätes besteht darin, daß Sie einen **dunkelblauen** Bildschirmhintergrund mit **hellblauem** Rand und **hellblauen** Zeichen erhalten sollten.

Wenn sich ihr COMMODORE 64 nicht wie beschrieben auf dem Bildschirm Ihres Fernsehgerätes oder Monitors meldet, überprüfen Sie Ihren Aufbau noch einmal anhand der folgenden Liste.

FEHLERSUCHE

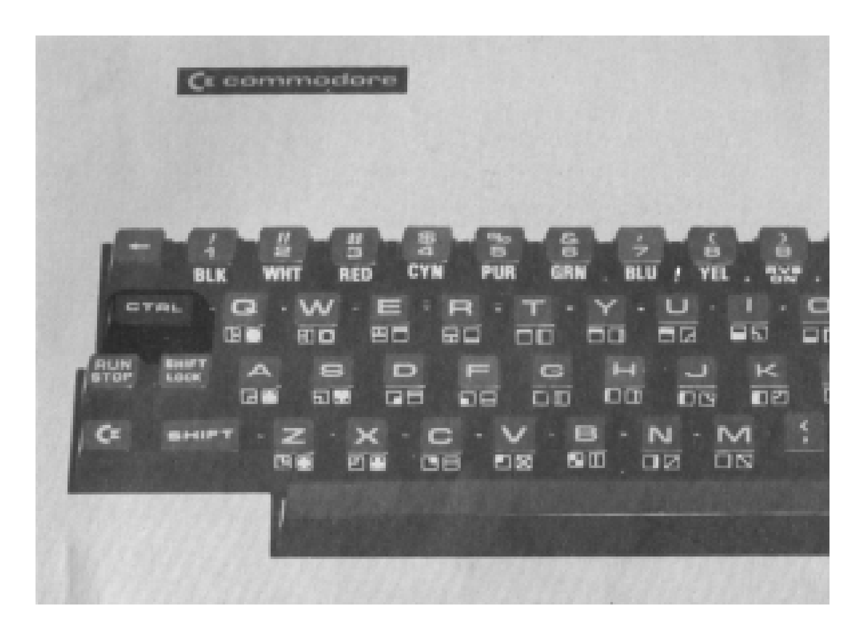
SYMPTOM	URSACHE	MASSNAHME
Rote Indikatorlampe am Computer brennt nicht	Computer nicht eingeschaltet	Schalter auf der rechten Seite des Computers muß in der Stellung "ON" sein
	Keine Verbindung zum Netz	Überprüfen Sie Netzstecker, Gerätestecker und gegebenenfalls auch die Steckdose
	Sicherung defekt	Bitten Sie Ihren Fachhändler die Sicherung auszutauschen
Keine Bildschirmanzeige	Falscher Kanal am Fernsehapparat gewählt	Überprüfen Sie den Kanalwählschalter am Fernsehapparat
	Antennen - bzw. Videokabel nicht eingesteckt oder defekt	Betreffendes Kabel überprüfen
Zeichenwirrwarr auf dem Bildschirm bei eingestecktem Modul	Modul nicht korrekt eingesteckt	Gerät ausschalten und Sitz des Moduls im Steckplatz überprüfen
Schlechte Farben	Fernsehgerät nicht korrekt eingestellt	Farb- und Kanaleinstellung am Fernsehgerät überprüfen
Starkes Rauschen	Lautstärke am Fernseh- gerät zu hoch eingestellt	
Bild o. k., aber kein Ton	Lautstärke am Fernsehgerät zu niedrig eingestellt	
	Audioausgang nicht mit dem Eingang des Verstärkers verbunden	Verbinden Sie den Audioausgang des Computers mit dem AUX-Eingang Ihres Verstärkers und stellen Sie den Input-Wählknopf auf AUX

DER "CURSOR"

Das helle blinkende Quadrat unter der READY-Meldung wird als CURSOR bezeichnet. Es zeigt die Position an, in der ein von der Tastatur eingegebener Buchstabe auf dem Bildschirm erscheint. Geben Sie einen Text ein und beobachten Sie, wie der Cursor sich über den Bildschirm bewegt und jeweils durch die von Ihnen eingegebenen Zeichen ersetzt wird.

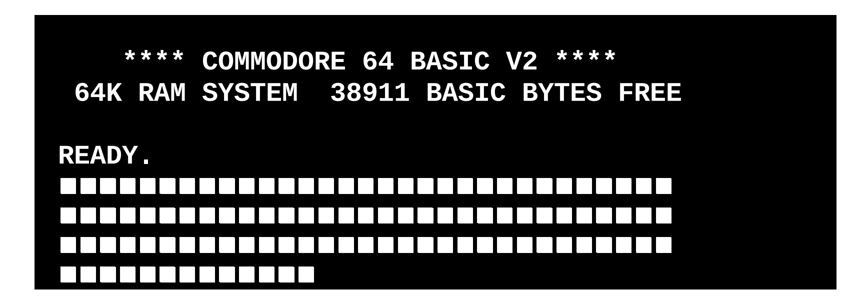
EINSTELLUNG DER BILDSCHIRMFARBEN

Die Einstellung der Farben geschieht anhand eines Musters, das sich mit einfachen Mitteln auf dem Bildschirm Ihres Fernsehgerätes oder Monitors erzeugen läßt.
Sie benutzen dazu die **<CTRL>**-Taste auf der linken Seite der Tastatur; der Name ist abgeleitet von ConTRoL - kontrollieren. Die **<CTRL>**-Taste wird stets zusammen mit einer anderen Taste angewendet, wobei stets beide Tasten **zugleich** gedrückt werden müssen. Sie schalten hierbei den Computer in einen bestimmten Modus um, in dem er in der Regel bleibt, bis Sie ihn wieder umschalten.
Drücken Sie nun die **<CTRL>**-Taste zusammen mit der **<9>**-Taste. Sie werden nach dem Loslassen zuerst keine Wirkung feststellen. Wenn Sie jetzt jedoch ein Zeichen eintippen, so werden Sie feststellen, daß es auf dem Bildschirm in Revers-Darstellung erscheint, d. h. Bildschirm- und Zeichenfarbe sind vertauscht.
Drücken Sie nun auf die **<SPACE>**-Taste. Wenn Sie alles richtig gemacht haben, so erscheint ein heller Balken auf dem Bildschirm, der solange wächst, wie Sie die **<SPACE>**-Taste niedergedrückt lassen.



Wenn Sie die **<CTRL>**-Taste zusammen mit einer anderen Ziffern-Taste drücken, schalten Sie die Zeichenfarbe um. Schalten Sie nun mit Hilfe der **<8>**-Taste die Zeichenfarbe **gelb** ein. Drücken Sie nun wieder auf die **<SPACE>**-Taste! Was passiert? Sie können einen beliebig langen Balken von gelber Farbe erzeugen.
Erzeugen Sie auf diese Weise ein Farbmuster auf dem Bildschirm und regeln Sie die Farb- und Helligkeitskontrollen Ihres Fernsehgerätes oder Monitors so ein, daß Sie möglichst kräftige reine Farben erhalten, die mit den Bezeichnungen auf den Farbtasten Ihres COMMODORE 64 übereinstimmen.

Ihr Bildschirm sollte etwa wie folgt aussehen:



Ihr System ist nun korrekt eingestellt.
Die folgenden Kapitel geben Ihnen nun eine Einführung in die Programmiersprache BASIC. Sie können aber auch fertige Programme verwenden, für deren Betrieb Sie keine Programmierkenntnisse benötigen.
Diesen Programmpaketen sind ausführliche Anleitungen beigefügt, die Ihnen die Anwendung erleichtern; es empfiehlt sich jedoch, die ersten Kapitel dieses Handbuches durchzuarbeiten, um mit den grundlegenden Eigenschaften Ihres neuen Systems vertraut zu werden.

KAPITEL 2

JETZT GEHT ES ENDLICH LOS

- **Die Tastatur**
- **Laden und Speichern von Programmen**
- **Der Befehl PRINT**
- **Rechenoperationen**

DIE TASTATUR

Nehmen Sie sich nun ein paar Minuten Zeit, um mit der Tastatur vertraut zu werden, da sie das wichtigste Instrument zur Eingabe von Informationen in den COMMODORE 64 darstellt.
Im großen und ganzen stimmt die Tastatur mit der einer Schreibmaschine überein. Es gibt jedoch einige zusätzliche Tastenfunktionen, die im folgenden kurz beschrieben werden. Eine genaue Beschreibung folgt in den späteren Kapiteln.



<RETURN>

Die **<RETURN>**-Taste veranlaßt den Computer, die von Ihnen eingetippte und auf dem Bildschirm sichtbare Information in seinen Speicher einzulesen.

<SHIFT>

Die **<SHIFT>**-Taste entspricht in etwa der Groß/Kleinschrift-Umschalttaste bei der Schreibmaschine. Im "Groß/Kleinschrift-Modus", der weiter unten erklärt wird, erreicht man mit der **<SHIFT>**-Taste die Großbuchstaben; im "Gro8schrift/Grafik-Modus" erhält man die Grafik-Symbole auf der rechten Seite der Tasten.
Bei mehrfach belegten Funktionstasten (z. B. **<CLR/HOME>**), wird die oben auf der Taste stehende Funktion angesprochen (im angegebenen Fall die Funktion CLR).

Edieren

Jeder macht mal einen Fehler, und der COMMODORE 64 berücksichtigt das. Er hat eine Reihe von Tastenfunktionen die Ihnen das Korrigieren sehr erleichtern.

<CRSR>

Sie erinnern sich noch, wir haben das blinkende Rechteck, das angibt, wo wir uns gerade auf dem Bildschirm befinden, CURSOR genannt. Die **<CRSR>**-Tasten (abgeleitet von CuRSoR) geben uns die Möglichkeit, den Cursor in jede gewünschte Bildschirmposition zu bringen. Mit der linken der beiden **<CRSR>**-Tasten, der **<⇧CRSR⇩>**-Taste läßt sich der Cursor nach unten und (zusammen mit der **<SHIFT>**-Taste) nach oben bewegen. Die rechte Cursor-Steuertaste, die **<⇦CRSR⇨>**-Taste erlaubt Bewegungen des Cursors nach rechts und (mit **<SHIFT>**) nach links. Die **<CRSR>**-Tasten sind mit der sogenannten "Repeat"-Funktion versehen, d. h., der Cursor bewegt sich solange, wie Sie die Taste gedrückt halten.

<INST/DEL>

(Die Bezeichnungen sind abgeleitet von DELete = löschen und INSerT = einfügen)

Drücken Sie beim Schreiben eines Wortes die **<INST/DEL>**-Taste, so wird das zuletzt geschriebene Zeichen gelöscht. Löschen Sie ein Zeichen aus einer bereits bestehenden Zeile heraus, so wird das vor dem Cursor stehende Zeichen gelöscht, und der rechts vom Cursor befindliche Teil der Zeile um eine Position nach links verschoben.
Die **<SHIFT>**-Taste zusammen mit der **<INST/DEL>**-Taste erlaubt das Ein- fügen von Zeichen in bereits geschriebene Zeilen. Bewegen Sie den Cursor mit Hilfe der **<CRSR>**-Tasten an die Stelle, an der ein Zeichen fehlt. Drücken Sie nun **<SHIFT>** und **<INST/DEL>**; das Zeichen auf dem sich der Cursor befindet und damit der gesamte rechts vom Cursor stehende Zeilenabschnitt werden um eine Position nach rechts verschoben. Jetzt brauchen Sie nur noch das fehlende Zeichen einzutippen.

<CLR/HOME>

Durch Drücken der <**CLR/HOME>**-Taste bringen Sie den Cursor in die sog. "HOME-POSITION", das ist die linke obere Ecke des Bildschirms. Die **<SHIFT>**-Taste zusammen mit der **<CLR/HOME>**-Taste löscht den Bildschirm **und** bringt den Cursor in die Home-Position.

<RESTORE>

(Abgeleitet von RESTORE = wiederherstellen)

Drücken Sie die **<RESTORE>**-Taste zusammen mit der **<RUN/STOP>**-Taste, so können Sie Ihren Computer in den "Ausgangszustand" zurückbringen. Was damit gemeint ist, wird später näher erläutert.

Die Funktionstasten

Die Funktionstasten auf der rechten Seite der Tastatur können im Rahmen eines Programms frei mit Bedeutungen belegt werden. Dabei benutzt man die Tatsache, daß den 4 Funktionstasten die ASCII-Codes 133-136 (mit der **<SHIFT>**-Taste zusammen die ASCII-Codes 137-140) entsprechen. (Siehe in diesem Zusammenhang S.135 ff.)



<CTRL>

Die **<CTRL>**-Taste (abgeleitet von ConTRoL), dient etwa zum Einschalten verschiedener Farben, oder dem Ein- und Ausschalten des Revers- Modus. Die **<CTRL>**-Taste muß stets zusammen mit der entsprechenden Taste (z. B. der Farbtaste) gedrückt werden.

<RUN/STOP>

Die **<RUN/STOP>**-Taste dient dazu, den Ablauf eines Basic-Programms zu unterbrechen. Der Computer erhält den Befehl, einen Vorgang zu STOPpen. Zusammen mit der **<SHIFT>**-Taste wird ein Programm vom Band geladen (und gestartet).

<C=> Die "COMMODORE"-Taste

Die Taste **<C=>** mit dem COMMODORE-Symbol hat eine ganze Reihe von Funktionen. Als erstes können Sie mit ihrer Hilfe die linksstehenden Zeichen auf den Vorderseiten der Tasten erreichen (die rechtsstehenden mit der **<SHIFT>**-Taste).
Wenn Sie den Computer einschalten, so ist er automatisch im Großschrift/Grafik-Modus, d. h. Sie können in Großbuchstaben schreiben und alle Grafikzeichen benutzen. Drücken Sie gleichzeitig die **<C=>**-Taste und die **<SHIFT>**-Taste, so wird eine andere Kombination von Zeichensätzen eingeschaltet, Sie kommen in den Groß/Kleinschrift-Modus. Die Bezeichnung ist eigentlich nicht ganz korrekt, da Sie auch in diesem Modus noch einige Grafikzeichen zur Verfügung haben; Sie können mit Hilfe der **<C=>**-Taste die links auf der Tastenvorderseite stehenden Grafikzeichen erreichen. Die Buchstabentasten sind jetzt mit Kleinbuchstaben belegt, die Großbuchstaben erreichen Sie mit Hilfe der **<SHIFT>**-Taste. Eine weitere Funktion der **<C=>**-Taste besteht darin, daß Sie mit ihr (in Kombination mit den Zifferntasten) weitere 8 Zeichenfarben einschalten können. Die Zuordnung der Farben zu den Tasten ist im Kapitel 5 aufgelistet.

WIE VERSETZE ICH DEN COMPUTER WIEDER IN DEN "NORMALZUSTAND"?

Wenn Sie noch etwas auf dem Bildschirm haben, z. B. die Farbbalken, so drücken Sie die **<SHIFT>**- und die **<CLR/HOME>**-Taste, Sie erhalten einen "sauberen" Bildschirm mit dem Cursor in der linken oberen Ecke.
Drücken Sie nun die **<C=>**-Taste zusammen mit der **<7>**-Taste**;** hiermit schalten Sie die "normale" hellblaue Zeichenfarbe wieder ein. Falls Sie sich noch im Revers-Modus befinden (den brauchen Sie ja zur Herstellung der Farbbalken), so müssen Sie diesen noch durch **<CTRL> <0>** ausschalten.

Wir benutzen im folgenden die Abkürzung **<A> <B>** für: Die Tasten **<A>** und **<B>** werden gleichzeitig gedrückt).

TIP:
Sie haben sich bei der Einstellung der "Normalwerte" viel Arbeit gemacht. Es gibt einen wesentlich einfacheren Weg. Drücken Sie nur die Kombination:

<RUN/STOP> <RESTORE>

Dieser Befehl setzt viele wichtige Register wieder auf Ihre Anfangswerte zurück, läßt jedoch ein Programm, das sich im Speicher befindet, unberührt.
Wenn Sie den Computer wieder in den Zustand versetzen wollen, in dem er direkt nach dem Einschalten war, so müssen Sie die Zeile SYS 64738 eintippen und die **<RETURN>**-Taste drücken. Gehen Sie jedoch vorsichtig mit diesem Befehl um; das Programm und alle sonstigen Speicherinhalte werden gelöscht.

LADEN UND ABSPEICHERN VON PROGRAMMEN

Eine der wichtigsten Eigenschaften Ihres COMMODORE 64 besteht darin, daß Sie Programme auf Band oder Diskette abspeichern können um sie bei späterem Bedarf wieder in den Speicher des Computers einzulesen.
Stellen Sie sicher, daß die Datasette bzw. das Diskettenlaufwerk korrekt an den Computer angeschlossen sind.

Das Laden von gekauften Programmen

1. **MODULE:** Es gibt eine ganze Reihe von Spielen und anderen Programmen auf sogenannten Steckmodulen. Wenn Sie diese Module in die vorgesehene Position einstecken, vergessen Sie nicht: **VOR DEM EINSTECKEN DER MODULE STETS COMPUTER AUSSCHALTEN!!!**
   Wenn Sie diesen Ratschlag nicht befolgen, können Ihre Module beschädigt werden. Schalten Sie nun Ihren COMMODORE 64 ein und starten Sie das Programm mit dem in der Anleitung angegebenen Befehl oder der jeweiligen Starttaste.

2. **KASSETTEN:** Legen Sie die Kassette in die Datasette (den zugehörigen Kassettenrecorder) ein und spulen Sie gegebenenfalls bis zum

Anfang zurück. Tippen Sie dann in den Computer LOAD (**<RETURN>**-Taste nicht vergessen!) ein; er wird Ihnen mit der Anweisung antworten PRESS PLAY ON TAPE. Folgen Sie nun dieser Anweisung und drücken die mit PLAY bezeichnete Taste an der Datasette. Die Schrift auf dem Bildschirm und auch der Cursor verschwinden nun, das Band beginnt zu laufen und schließlich meldet sich der Computer wieder mit: FOUND (PROGRAMMNAME). Auf Drücken der **<C=>**-Taste hin wird das Programm in den Speicher des COMMODORE 64 geladen. Sie können den Ladevorgang jedoch jederzeit durch Drücken der **<RUN/STOP>**-Taste unterbrechen.

3. **DISKETTE:** Öffnen Sie das Laufwerk, orientieren Sie die Diskette so, daß das Etikett nach oben und zu Ihnen hin weist und schieben Sie sie dann vorsichtig in den Schlitz. Achten Sie auf die kleine Einkerbung an der Seite der Diskette; wenn Sie alles richtig gemacht haben, muß sich diese Einkerbung (die auch mit einem Stück Klebeband zugeklebt sein kann), auf der **linken** Seite befinden. Wenn sich die Diskette im Laufwerk befindet, schließen Sie vorsichtig die Klappe und tippen in den Computer ein LOAD"PROGRAMMNAME",8 und drücken **<RETURN>**. Sie werden hören, daß das Laufwerk arbeitet und auf dem Bildschirm erscheint:



Wenn READY und der Cursor auf dem Bildschirm erschienen sind, können Sie durch Eintippen von RUN und Drücken der **<RETURN>**-Taste das Programm starten.

Das Laden von Programmen von Band

Wenn Sie ein Programm von Band laden wollen, das Sie vorher unter dem Namen PROGRAMMNAME abgespeichert haben, so spulen Sie das Band zurück und tippen ein:

```
LOAD "PROGRAMMNAME"
```

Wenn Sie sich nicht mehr an den Namen des Programms erinnern können, so tippen Sie nur LOAD ein und drücken die **<RETURN>**-Taste. In diesem Fall wird einfach das erste Programm auf dem Band geladen. Der Computer meldet sich dann mit:

```
PRESS PLAY ON TAPE
```

Wenn Sie nun dieser Aufforderung folgen und die PLAY-Taste der Datasette drücken, so verschwinden Schrift und Cursor vom Bildschirm und der Computer sucht das von Ihnen gewählte Programm.
Wenn er es auf dem Band gefunden hat, meldet er sich mit:

```
FOUND PROGRAMMNAME
```

Um das Programm in den Rechnerspeicher zu laden, müssen Sie jetzt die **<C=>-**Taste drücken. Wenn der Ladevorgang beendet ist, erscheinen dann die READY-Meldung und der Cursor wieder auf dem Bildschirm.

Das Laden von Programmen von Diskette

Wenn Sie Programme von der Diskette laden wollen, müssen Sie ähnlich wie bei der Kassette vorgehen. Tippen Sie ein:

```
LOAD "PROGRAMMNAME",8
```

Nach dem Drücken der **<RETURN>**-Taste erhalten Sie folgenden Kommentar auf dem Bildschirm:

```
SEARCHING FOR PROGRAMMNAME
LOADING
READY.
```



Bem.:
Wenn Sie ein Programm in den Speicher des Computers laden, so wird die gesamte Information, die vorher darin enthalten war, gelöscht. Überprüfen Sie daher, ob Sie das Programm, an dem Sie gearbeitet haben, abgespeichert haben, bevor Sie das neue laden.

DAS SPEICHERN VON PROGRAMMEN AUF BAND

Wenn Sie ein Programm geschrieben haben und Sie wollen es auf Kassette abspeichern, so tippen Sie ein (wobei PROGRAMMNAME für einen von Ihnen frei wählbaren Programmnamen steht):

```
SAVE "PROGRAMMNAME"
```

Der Programmname kann bis zu 16 Zeichen lang sein. Nach dem Drücken der **<RETURN>**-Taste meldet sich der Computer mit:

```
PRESS PLAY AND RECORD ON TAPE
```

Drücken Sie daraufhin gleichzeitig die PLAY- und die RECORD-Taste an Ihrem Kassettenrecorder. Die Schrift und der Cursor verschwinden vom Bildschirm bis das Programm vollständig abgespeichert ist, worauf der Computer wieder "ansprechbar" ist.

Das Abspeichern von Programmen auf Diskette

Fast noch einfacher ist das Speichern von Programmen auf Disketten. Tippen Sie ein:

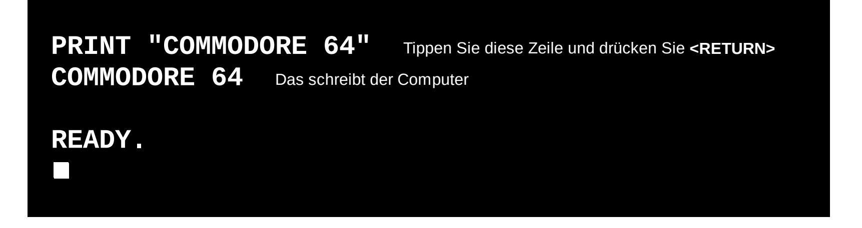
```
SAVE "PROGRAMMNAME",8
```

Die "8" ist die Gerätenummer des Diskettenlaufwerks und informiert den Computer darüber, daß Sie Ihr Programm auf Diskette abspeichern wollen.
Nach dem Drücken der **<RETURN>**-Taste meldet sich der COMMODORE 64 wieder mit:

```
SAVING PROGRAMMNAME
READY.
■
```

DER PRINT-BEFEHL UND DAS RECHNEN MIT DEM COMMODORE 64

Da Sie nun wissen, wie man ein Programm abspeichert, wollen wir darangehen eins zu schreiben.
Tippen Sie folgendes ein und beachten Sie die Kommentare in der Abbildung.

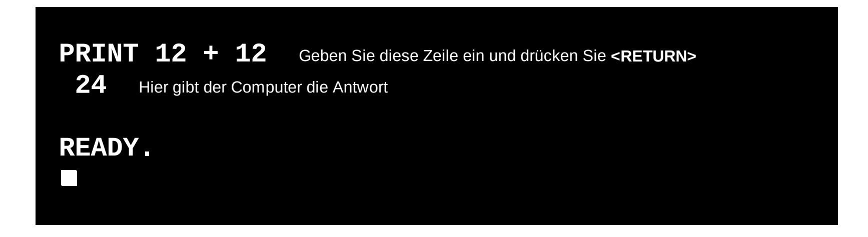


Wenn Sie beim Eintippen einen Fehler gemacht haben, so benutzen Sie die **<INST/DEL>**-Taste um das Zeichen, das links vom Cursor steht zu löschen. Sie können auf diese Weise beliebig große Abschnitte löschen.

Wir wollen uns nun einmal näher anschauen, was Sie eingetippt haben. Als erstes haben Sie einen Befehl formuliert, der den Computer anweist, den Text, der zwischen den Anführungszeichen steht, auf den Bildschirm zu schreiben. Der COMMODORE 64 erkennt den Befehl als solchen aber erst nach dem Drücken der **<RETURN>**-Taste, und er führt ihn dann auch erst aus, d. h. er schreibt COMMODORE 64 auf den Bildschirm.
Wenn der Computer sich mit

?SYNTAX ERROR

meldet, so haben Sie etwas falsch gemacht; vielleicht haben Sie sich vertippt, oder ein Anführungszeichen vergessen. Der Rechner ist sehr "pingelig" und erkennt die Befehle nur, wenn sie genau dem vorgegebenen Schema entsprechen. Aber haben Sie keine Angst vor dem Experimentieren; Sie können den Computer durch Eintippen falscher Befehle nicht beschädigen, und die beste Methode, BASIC zu lernen, besteht darin, alles auszuprobieren und zu beobachten was passiert.
Der PRINT-Befehl ist einer der wichtigsten und nützlichsten Befehle. Sie können mit seiner Hilfe alles was Sie wollen auf den Bildschirm bringen; eingeschlossen die Grafikzeichen oder die Resultate von Rechnungen. Probieren Sie doch einmal das folgende Beispiel aus. Löschen Sie dazu zuerst mit **<SHIFT> <CLR/HOME>** Bildschirm und tippen dann folgendes ein (denken Sie daran, daß Sie eine "1" und kein "I" tippen).



Was Sie hier ausgenutzt haben, ist die Tatsache, daß Sie den COMMODORE 64 als Rechenmaschine benutzen können. Sie müssen nur daran denken, daß Sie vor die von Ihnen gewünschte Rechnung einen PRINT-Befehl setzen. Sie können alle Rechenarten einsetzen, die auf Ihrem Computer verfügbar sind, so z. B. neben der Addition auch Subtraktion, Multiplikation, Division, das Rechnen mit Exponenten, Wurzeln, trigonometrischen Funktionen etc.
Vergleichen Sie was passiert, wenn Sie 12+12 in Anführungszeichen setzen!

Addition:

Die Addition wird mit Hilfe des (+)-Zeichens ausgeführt. In unserem Beispiel wurde die Rechnung 12 + 12 ausgeführt. Denken Sie daran, daß der Computer erst nach dem Drücken der **<RETURN>**-Taste Ihren Befehl ausführen kann.

Subtraktion

Zum Subtrahieren verwendet man das normale (-)-Zeichen.

```
PRINT 12 - 9   <RETURN>
 3
```

Multiplikation

Die Multiplikation wird durch das "Sternchen" (*) durchgeführt.

```
PRINT 12 * 12   <RETURN>
 144
```

Division

Zur Division benutzt man den Schrägstrich (/).

```
PRINT 144/12   <RETURN>
 12
```

Exponentiation

In ähnlicher Weise wie die anderen Rechenoperatoren benutzen Sie das Zeichen (↑), um eine Zahl in eine bestimmte Potenz zu erheben.

```
PRINT 12 ↑ 5   <RETURN>
 248832
```

Das ist die abgekürzte Schreibweise für:

```
PRINT 12 * 12 * 12 * 12 * 12   <RETURN>
 248832
```

TIP:
Es gibt eine ganze Anzahl von Abkürzungen für BASIC-Befehle, die im Anhang D systematisch zusammengestellt sind. Eine wichtige Abkürzung ist das ? für den Befehl PRINT

Probieren Sie diese Abkürzung doch gleich einmal im nächsten Beispiel aus, in dem verschiedene Rechenoperationen im Zusammenhang mit einem PRINT-Befehl ausgeführt werden.

```
? 3 + 5 - 7 + 2    Dieses ? ersetzt das Wort PRINT
 3
```

Bis jetzt haben wir nur kleine Zahlen und einfache Beispiele verwendet; aber der COMMODORE 64 kann wesentlich mehr. Nehmen Sie die nächste Aufgabe als Beispiel und addieren Sie eine Reihe großer

Dezimalbrüche. Vergessen Sie aber nicht, statt der manchmal gebräuchlichen Kommas Punkte zu verwenden.

```
? 123.45 + 345.78 + 7895.687
 8364.917
```

Das sieht ja schon ganz gut aus, aber schauen Sie sich erst die nächste Aufgabe an:

```
? 12123123.45 + 345.78 + 7895.687
 12131364.9
```

Wenn Sie sich die Zeit nehmen und das Beispiel von Hand durchrechnen, kommen Sie zu einem anderen Ergebnis. Woher kommt das?
Die Rechengenauigkeit Ihres COMMODORE 64 ist zwar groß, aber dennoch begrenzt. Er rechnet mit zehn Stellen, von denen er jedoch nur neun anzeigt.
In unserem Beispiel hat der Computer deshalb "gerundet", um in seinem Genauigkeitsbereich zu bleiben. Wie in der Mathematik üblich, rundet er von der 1 bis zur 4 ab, von der 5 bis zur 9 auf.
Zahlen zwischen 0.01 und 999999999 werden im Standardformat ausgegeben. Zahlen, die außerhalb dieses Bereiches liegen, werden in der wissenschaftlichen Notation angezeigt.
Die wissenschaftliche Notation ist eine Methode, sehr große bzw. sehr kleine Zahlen als Potenzen von 10 darzustellen.
Tippen Sie folgendes Beispiel ein:

```
? 123000000000000000
 1.23E+17
```

Wie Sie sehen kann die sehr große Zahl, die von Ihnen hinter den PRINT-Befehl geschrieben wurde, in knapper Form als 1.23*10↑17 dargestellt werden.

Es gibt jedoch selbst in dieser Darstellung Grenzen, zwischen denen die Zahlen liegen müssen, wenn sie vom Computer verarbeitet werden sollen. Diese Grenzen sind:

Größte Zahl: 1.70141183 E+38
Kleinste Zahl: 2.93873588 E-39 (was darunter liegt, wird als 0 behandelt)

KOMBINATION VERSCHIEDENER RECHENOPERATIONEN

Vielleicht haben Sie einige Beispiele eingegeben, die verschiedene Arten von Rechenoperationen enthielten und haben nicht das erwartete Ergebnis enthalten. Sie müssen deshalb die Systematik kennenlernen, nach der der Computer bei Auswerten von Rechenausdrücken vorgeht.
Überlegen Sie sich einmal, welches Ergebnis die folgende Rechnung hat:

20 + 8 / 2

Je nach Vorgehensweise kommt man auf 14 oder 24. Ihr COMMODORE 64 bekommt 24 heraus. Der Grund dafür ist, daß er die Division **vor** der Addition durchführt. Hier ist eine Rangfolge der Rechenoperation; die weiter obenstehenden Operationen werden **vor** den weiter untenstehenden ausgeführt.

1. Rang	-	Das Minuszeichen **als Vorzeichen** einer Zahl
2. Rang	↑	Die Exponentiation (Erheben in eine Potenz)
3. Rang	* /	Multiplikation und Division
4. Rang	+ -	Addition und Subtraktion

Überprüfen Sie das angegebene Ergebnis von 24 an Hand dieser Regeln. Sie können die Reihenfolge der Rechenoperationen dadurch beeinflussen, daß Sie Klammern setzen. Die Auswirkungen, die das Setzen von Klammern hat, können Sie am nächsten Beispiel studieren.
Geben Sie an Ihrem Computer das folgende Rechenbeispiel ein:

```
? 35 / 5 + 2
 9
```

D. h. der Computer dividiert 35 durch 5 (= 7) und addiert zum Ergebnis 2 dazu (= 9). Setzen Sie nun die Summe 5 + **2** in Klammern:

```
? 35 / (5 + 2)
 5
```

Nun hat der COMMODORE 64 als erstes die Rechenoperation in der Klammer ausgeführt (= 7) und 35 durch das Ergebnis dividiert (= 5).
Im nächsten Beispiel berechnet der Computer als erstes die beiden Klammern und multipliziert die Ergebnisse.

```
? (12 + 9) * (6 + 1)
 147
```

DER PRINT-BEFEHL MIT UND OHNE VERWENDUNG VON ANFÜHRUNGSZEICHEN

Bis jetzt hat der Computer immer nur das Ergebnis einer Rechnung auf dem Bildschirm ausgegeben. Durch den Einsatz von Anführungszeichen können Sie erreichen, daß die Aufgabenstellung **und** das Ergebnis auf dem Bildschirm erscheinen.
Denken Sie daran, das alles was in Anführungszeichen gesetzt wird, genauso auf dem Bildschirm wiedergegeben wird, während von einer Rechnung, die nicht in Anführungszeichen gesetzt wurde, nur das Ergebnis erscheint. Diese Eigenschaften des COMMODORE 64 werden im nächsten Beispiel ausgenutzt.
Das Semikolon hinter dem zweiten Anführungszeichen bewirkt, daß das Ergebnis der Rechnung unmittelbar hinter die Rechnung geschrieben wird.

```
? "5 * 9 = "; 5 * 9
5 * 9 = 45
```

Was wir da eingetippt haben, sieht auf den ersten Blick ein bißchen umständlich aus, wird Ihnen aber nach kurzem Nachdenken einleuchten. Der erste Teil: "5 * 9 =" druckt die Aufgabenstellung auf dem Bildschirm aus und der Teil hinter dem Semikolon (;) bildet die eigentliche Rechenvorschrift, deren Ergebnis der Computer ausdruckt.
Es ist nicht unwichtig, mit welchem Zeichen Sie die beiden Teile des PRINT-Befehls trennen. Probieren Sie doch einmal aus, was passiert, wenn Sie ein Komma an die Stelle des Semikolons setzen...
Während bei Benutzung des Semikolons das Ergebnis direkt hinter der Aufgabenstellung ausgedruckt wird, entsteht bei Setzen eines Kommas ein Zwischenraum. Um genauer zu untersuchen, was das Komma verursacht, tippen Sie bitte folgendes Beispiel ein:

```
? 2,3,4,5,6  <RETURN>
 2         3         4         5
 6
```

Ihr Commodore 64 verteilt also offensichtlich jeweils vier Zeichen gleichmäßig über eine Zeile. Überzählige Zeichen "schiebt" er in die nächste Zeile.
Der Grund liegt darin, daß der Bildschirm des COMMODORE 64 so organisiert ist, daß jede Zeile aus 4 Abschnitten à 10 Zeichen besteht. Das Komma entspricht einem Tabulator, der jeweils an den Anfang des nächsten Abschnittes springt, auch wenn dieser sich in der nächsten Zeile befindet. Es ist wichtig, sich den Unterschied zwischen Komma und Semikolon zu merken, da er bei der Ausgabe von Ergebnissen auf dem Bildschirm nützlich angewendet werden kann.

KAPITEL 3

GRUNDLAGEN DES PROGRAMMIERENS IN BASIC

- **GOTO**
- **Tips zum Edieren**
- **IF ... THEN**
- **FOR ... NEXT Schleifen**

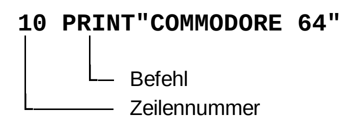
DIE NÄCHSTEN SCHRITTE

Bis jetzt haben wir uns damit begnügt, einfache Operationen auszuführen, indem wir einzelne Befehlszeilen in den Rechner eingegeben haben. Nach dem Drücken der **<RETURN>**-Taste wurden die in dieser Zeile enthaltenen Befehle dann vom Computer ausgeführt. Diese Art mit dem COMMODORE 64 zu rechnen nennt man den DIREKT-Modus.
Aber unser Computer kann mehr! Er kann eine große Zahl von Befehlszeilen nacheinander ausführen; d. h., er kann ein PROGRAMM abarbeiten.
Um Ihnen zu zeigen, wie einfach es ist, ein Programm für Ihren COMMODORE 64 zu schreiben, hier ein Beispiel:
Löschen Sie erst den Bildschirm mit **<SHIFT> <CLR/HOME>**. Tippen Sie dann NEW ein und drücken die **<RETURN>**-Taste; damit wird der Programmspeicher Ihres Computers von allem geleert, was Sie im Verlauf Ihrer Experimente eventuell hineingeschrieben haben.
Tippen Sie nun genau den folgenden Text ab, und vergessen Sie nicht, nach Eingabe jeder Zeile **<RETURN>** zu drücken.



Tippen Sie nun RUN ein und drücken die **<RETURN>**-Taste. Was passiert? Der Bildschirm wird lebendig. Wenn Sie genug haben, unterbrechen Sie das Programm durch Drücken der **<RUN/STOP>**-Taste.
Wir haben in diesem kurzen Beispiel schon einige Dinge benutzt, die für das Programmieren von allgemeiner Wichtigkeit sind.
Beachten Sie, daß wir vor jeden Befehl eine Zeilennummer geschrieben haben. Diese Zahlen geben dem Computer an, in welcher Reihenfolge er Ihre Befehle abarbeiten soll. Soll während des Programmablaufs zu einem bestimmten Punkt zurückgesprungen werden (das macht im vorliegenden Beispiel der Befehl GOTO 10), so dienen die Zeilennummern als Markierungspunkte, auf die man sich bezieht. Als Zeilennummern können alle ganzen Zahlen zwischen 0 und 63999 gewählt werden.

10 PRINT"COMMODORE 64"



```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
BREAK IN 10
READY.
■
```

Es entspricht guter Programmierpraxis, die Zeilen in Zehnerabständen zu numerieren - das vereinfacht das nachträgliche Einfügen von Programmzeilen. Neben PRINT kam in unserem Programm noch ein anderer BASIC-Befehl vor, nämlich GOTO. Damit geben Sie dem Computer die Anweisung, in die hinter GOTO angegebene Zeile zu springen und dort mit der Programmausführung fortzufahren.

```
┌──>  10 PRINT "COMMODORE 64"
│
└───  20 GOTO 10
```

Unser Programm läuft also folgendermaßen ab: In Zeile 10 wird Befehl zum Ausdrucken des Textes auf dem Bildschirm gegeben; das Programm geht dann in die Zeile 20 und erhält dort den Befehl, in die Zeile10 zurückzugehen. Damit fängt das Spiel von vorne an, und Ihr COMMODORE 64 wird nicht aufhören, seinen Namen auf den Bildschirm zu drucken, bis Sie ihn mit der **<RUN/STOP>**-Taste unterbrechen.
Tippen Sie nach der Programmunterbrechung LIST ein und drücken die **<RETURN>**-Taste. Sie sehen, ihr Programm befindet sich noch unverändert im Speicher des Computers. Beachten Sie, daß er das "?" in ein "PRINT" übersetzt hat. Das Programm kann nun geändert, abgespeichert oder wieder gestartet werden.
Beim Direkt-Modus (ohne Zeilennummern) ist das anders: wenn die Befehlszeile nicht mehr auf dem Bildschirm steht, ist sie verloren, und kann auch nicht mehr mit LIST zurückgeholt werden.

TIPS ZUM EDIEREN EINES PROGRAMMS

Wenn Sie in einer Programmzeile einen Fehler machen, gibt es einige Methoden, diesen zu korrigieren.

1. Sie können die Zeile noch einmal schreiben; die alte Zeile wird dann automatisch durch die neue, fehlerfreie ersetzt.
2. Wenn Sie nur die Nummer der fehlerhaften Zeile eintippen und **<RETURN>** drücken, so wird diese Zeile gelöscht.
3. Sie können die Zeile unter Benutzung der Cursor-Steuertasten und der Ediertasten beliebig verändern. Nach dem Drücken der **<RETURN>**-Taste wird dann die korrigierte Zeile übernommen.

Nehmen wir einmal an, Sie hätten sich in einer Programmzeile vertippt. Bewegen Sie nun den Cursor mit Hilfe der **<⇧CRSR⇩>**-Taste und der **<SHIFT>**- Taste in die fehlerhafte Zeile. Bringen Sie nun den Cursor unter Benutzung der **<⇦CRSR⇨>-**Taste und der **<SHIFT>**-Taste an die Stelle, wo das fehlerhafte Zeichen steht. Tippen Sie nun das richtige Zeichen ein, und drücken die **<RETURN>**-Taste; damit wird die richtige Zeile in den Programmspeicher übernommen, wovon Sie sich durch das Auslisten des Programms überzeugen können.
Müssen Sie ein Zeichen an einer Stelle einflicken, so fahren Sie wieder mit den Cursor-Steuertasten an die Stelle, wo Sie Platz für das zusätzliche Zeichen brauchen und tippen Sie **<SHIFT> <INST/DEL>** ein. Dadurch erzeugen Sie die benötigte Leerstelle. Nach dem Drücken der **<INST/DEL>**-Taste allein wird das vor dem Cursor stehende Zeichen gelöscht, ohne daß eine Lücke zurückbleibt.
Sie können die Zeilen in beliebiger Reihenfolge ändern; an den Zeilennummern erkennt Ihr Computer, wo er die korrigierten Zeilen einzuordnen hat.
Ändern Sie nun die Zeile 10 Ihres Programms so ab, daß sie der nachstehenden Zeile entspricht. Beachten Sie, daß am Ende der Zeile ein Semikolon (;) steht.

```
10 PRINT"COMMODORE ";
```

Starten Sie nun das Programm; nach dem Unterbrechen des Programms mit der **<RUN/STOP>**-Taste muß der Bildschirm etwa so aussehen, wie die Abbildung auf S. 35 wiedergibt.

VARIABLE

Einer der wichtigsten Begriffe in allen Programmiersprachen ist der der Variablen. Er ist deshalb wichtig, ihre Eigenschaften und Möglichkeiten kennenzulernen.

```
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
```

Stellen Sie sich vor, daß Ihr Computer eine Reihe von Schachteln enthält, die Zahlen oder Worte enthalten. Diese Schachteln tragen Namen, die Sie frei wählen können. Diese Namen nennen wir Variable; sie repräsentieren die Information, die zu einem bestimmten Zeitpunkt in der betreffenden Schachtel enthalten ist.
Nehmen wir folgendes Beispiel:

```
10 X% = 15
20 X = 23.5
30 X$ = "DIE SUMME VON X% + X = "
```

Der Computer ordnet die Variablen dann wie folgt zu:

```
X%   15
X    23.5
X$   DIE SUMME VON X% + X =
```

Eine Variable steht für einen Speicherplatz, der jedoch während eines Programmablaufs seinen Inhalt verändern kann. Wie Sie sehen kann man

einer Variablen eine ganze Zahl, einen Dezimalbruch oder auch einen Satz zuordnen.
Steht hinter dem Namen der Variablen ein %, so handelt es sich um einen Variablentyp, der nur ganze Zahlen repräsentieren kann (auch Integer-Variable genannt). Zulässige Integer-Variablennamen sind etwa die folgenden:

A%
X%
A1%
NM%

Steht hinter einem Variablennamen ein $, so handelt es sich um eine sogenannte "Stringvariable". Eine solche Variable kann für eine beliebige Aneinanderreihung von Zeichen (auch "Strings" genannt) stehen. Diese Strings können Buchstaben, Zahlen und auch Sonderzeichen enthalten. Beispiele gültiger Variablennamen für Strings sind z. B.

A$
X$
MI$

Dezimalbrüche (auch Fließkommazahlen oder "Floating-Point-Variablen" genannt), werden z. B. durch folgende Variablennamen repräsentiert:

A1
X
Y
MI

Beachten Sie, daß eine Variable aus mehr als zwei Zeichen bestehen kann; der Computer identifiziert die Variable jedoch anhand der ersten beiden Zeichen. Er kann also die beiden Variablen KOLLEGE und KONZERT nicht unterscheiden. Das erste Zeichen muß alphanumerisch (d. h., ein Buchstabe zwischen A und Z) und das zweite Zeichen ein Buchstabe oder eine Ziffer sein. Das letzte Zeichen kann dazu dienen, die Variable (wie oben beschrieben) näher zu charakterisieren (% = ganze Zahl; $ = String).
Weiterhin dürfen Variablennamen keine Worte enthalten, die im Basic feste Bedeutungen haben. So ist z. B. der Variablenname ROTOR nicht erlaubt, weil er den Basic-Begriff TO enthält. Eine Übersicht über diese reservierten Begriffe wird im Anhang D gegeben.
Tippen Sie nun das untenstehende Programm ein und starten Sie es mit RUN. (Vergessen Sie nicht, nach jeder Programmzeile <RETURN> zu drücken.)

```
NEW
10 X% = 15
20 X = 23.5
30 X$ = "DIE SUMME VON X% + X = "
40 PRINT "X% = "; X%, "X = ";X
50 PRINT X$; X% + X
```

Wenn Sie alles richtig gemacht haben, so sollte nach Beendigung des Programms folgendes auf dem Bildschirm stehen:

```
RUN
X% =  15  X =  23.5
DIE SUMME VON X% + X =  38.5
READY.
■
```

In den Zeilen 10 und 20 haben wir der Variablen X% die (ganze) Zahl 15 und der Variablen X den Dezimalbruch 23.5 zugeordnet. Der String X$ wurde in Zeile 30 definiert. In Zeile 40 haben wir verschiedene Formen der PRINT-Anweisung kombiniert, um die Namen und Werte der Variablen X und X% auszudrucken. Die Zeile 50 bewirkt, daß der String X$ und die Summe von X und X% auf den Bildschirm ausgegeben werden. Obwohl X Bestandteil aller verwendeten Variablennamen ist, kann der Computer auf Grund der Zusatzzeichen % und $ alle drei Variablen unterscheiden und sie mit verschiedenen Inhalten versehen.
Aber Sie können mit Variablen noch mehr anfangen; wenn Sie sie ändern, so wird der alte Inhalt der Variablen-"Schachtel" gegen den neuen ausgetauscht. Auf diese Weise können Sie Zuweisungen der folgenden Art benutzen:

```
X = X + 1
```

Als normaler algebraischer Ausdruck wäre diese "Gleichung" sinnlos, sie wird jedoch in Programmen außerordentlich häufig benutzt. Die Wirkung läßt sich so beschreiben: Nimm den Inhalt der Schachtel X, erhöhe den Inhalt um 1 und lege den neuen Inhalt wieder in die Schachtel mit der Aufschrift X.

IF ... THEN

Nachdem Sie nun wissen, wie man mit Variablen umgeht, wollen wir diese neuen Erkenntnisse gleich im nächsten Programm anwenden:

```
NEW
10 ZR = 0
20 ?"COMMODORE 64"
30 ZR = ZR + 1
40 IF ZR < 5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Wie Sie sehen, hat unser neuer Befehl IF ... THEN dazu geführt, daß nur noch eine begrenzte Anzahl von Zeilen ausgedruckt wird.
Stößt der Computer während der Programmausführung auf einen solchen Befehl, so prüft er, ob die hinter IF stehende Bedingung wahr ist. Wenn dies zutrifft, so wird der hinter THEN stehende Befehl ausgeführt. Sollte die hinter IF stehende Bedingung nicht mehr wahr sein, so führt der COMMODORE 64 den Befehl in der nächsten Zeile aus.
Eine ganze Reihe von Bedingungen können im Zusammenhang mit IF ... THEN verwendet werden:

SYMBOL	*BEDEUTUNG*
<	kleiner als
>	größer als
=	gleich
<>	ungleich
>=	größer oder gleich
<=	kleiner oder gleich

Sie können mit Hilfe dieser Bedingungen auf einfache aber wirkungsvolle Weise den Programmablauf kontrollieren.

```
      10 ZR = 0
┌──> 20 ?"COMMODORE 64"
│     30 ZR = ZR + 1
└─<─ 40 IF ZR < 5 THEN 20
       ↓
      50 END
```

Wir haben in unserem Beispielprogramm eine sogenannte "Schleife" konstruiert, die jedoch nur so lange durchlaufen wird, wie die Bedingung, die zwischen IF und THEN steht, erfüllt ist.
In Zeile 10 wird ZR (ZähleR) 0 gesetzt. Zeile 20 druckt unseren Text aus. In Zeile 30 wird der Wert von ZR bei jedem Durchlauf der Schleife um 1 erhöht. Der aktuelle Wert von ZR gibt also an, wie oft die Schleife abgearbeitet wurde.
In der Zeile 40 findet die Kontrolle statt. Wenn ZR kleiner als 5 ist, d. h., unser Text wurde bis dahin weniger als 5 mal ausgedruckt, dann geht das Programm zurück in die Zeile 20 und führt den PRINT-Befehl noch einmal aus. Nach dem 5. Durchlauf ist die Bedingung nicht mehr erfüllt, und das Programm geht weiter zur Zeile 50, wo der END-Befehl steht (der im vorliegenden Fall auch weggelassen werden könnte). Durch Änderung der oberen Grenze von ZR in der Zeile 40 können Sie jede beliebige Anzahl von Textzeilen erzeugen.
Wie wir noch sehen werden, hat der IF ... THEN-Befehl eine sehr große Zahl von Anwendungen.

FOR ... NEXT SCHLEIFEN

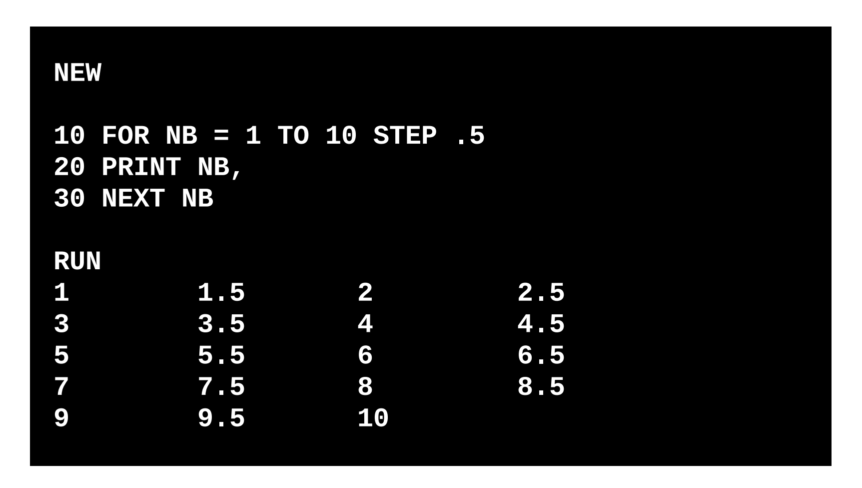
Der Effekt, den wir im vorigen Kapitel mit dem IF ... THEN Befehl erzielt haben, läßt sich auf leichtere Art mit der FOR ... NEXT Schleife realisieren. Tippen Sie bitte folgendes Programm ein und starten es mit RUN:

```
NEW

10 FOR ZR = 1 TO 5
20 PRINT "COMMODORE 64"
30 NEXT ZR

RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Sie erhalten den gleichen Bildschirmausdruck wie im letzten Kapitel, aber das Programm ist kürzer geworden.
In Zeile 10 wird ZR = 1 gesetzt. Zeile 20 bewirkt wieder das Ausdrucken unseres Textes, und in Zeile 30 wird der Wert von ZR um 1 erhöht. Der Ausdruck "NEXT" in Zeile 30 führt dazu, daß das Programm in die Zeile springt, wo das zugehörige "FOR" steht, nämlich in die Zeile 10. Die FOR ... NEXT Schleife wird so oft durchlaufen, bis der in Zeile 10 hinter TO stehende Wert erreicht ist.
Wenn Sie Ihrem Computer nichts anderes angeben, addiert er bei jedem Durchlauf den Wert 1; sie können jedoch auch jede andere Schritt- weite (auch Inkrement genannt) wählen. Tippen Sie dazu folgendes Beispiel ein:



```
NEW

10 FOR NB = 1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB

RUN
1        1.5       2         2.5
3        3.5       4         4.5
5        5.5       6         6.5
7        7.5       8         8.5
9        9.5       10
```

Durch die Angabe STEP .5 (die 0 im Dezimalbruch 0.5 kann weggelassen werden) wird bei jedem Durchlauf der Wert von NB um 0.5 erhöht, so daß die Zahlen 1, 1.5, 2, 2.5 etc. auf dem Bildschirm ausgegeben werden. Sie können die Zahl hinter dem Ausdruck "STEP" sogar negativ wählen. Probieren Sie doch einmal aus, was passiert, wenn Sie die Zeile 10 wie folgt ändern:

```
10 FOR NB = 10 TO 1 STEP -.5
```

KAPITEL 4

WEITERE BASIC-BEFEHLE

- Einleitung
- Bewegung auf dem Bildschirm - geschachtelte Schleifen
- INPUT
- GET
- Die Random-Funktion
- Zahlenratespiel
- Würfeln
- Zufallsmuster - die CHR$ - und die ASC-Funktion

EINLEITUNG

Die folgenden Kapitel richten sich an diejenigen Computerbenutzer, die bereits Erfahrung mit der Programmsprache BASIC haben. Es werden einige Methoden vorgestellt, die man zum Schreiben fortgeschrittener Programme benötigt.
Diejenigen, die erst mit dem Programmieren angefangen haben, werden einige Abschnitte vielleicht nicht sofort verstehen. Aber - nur Mut; für die beiden besonders unterhaltsamen Kapitel **"SPRITE-GRAFIK"** und **"TONERZEUGUNG"** haben wir uns auch ein paar Beispiele für den Anfänger einfallen lassen. Diese Beispiele werden Ihnen einen Eindruck davon vermitteln, welche faszinierenden Grafik- und Tonmöglichkeiten in Ihrem COMMODORE 64 stecken. Für diejenigen, die gar nicht genug kriegen können, haben wir im Anhang M noch eine Liste von BASIC-Lehrbüchern zusammengestellt.
Detaillierte Informationen speziell für den **COMMODORE 64** können Sie dem **PROGRAMMIERHANDBUCH** zum COMMODORE 64 entnehmen, das in Kürze bei Ihrem Fachhändler erhältlich sein wird.

BEWEGUNG AUF DEM BILDSCHIRM

Wir wollen nun noch einmal das üben, was wir bis jetzt gelernt haben, und noch einige neue Methoden dazu lernen. Nehmen Sie sich Mut und tippen Sie das nachstehende Programm ein. Das Bemerkenswerte an diesem Programm ist, daß der PRINT-Befehl zur Steuerung des Cursors benutzt wird. Wenn Sie im Programm z. b. den Cursor nach links bewegen wollen, so drücken Sie beim Eintippen die **<CRSR>**- und die **<SHIFT>**-Taste. Im Programmausdruck (auch "Listing" genannt) wird dies durch zwei senkrechte Streifen markiert. Das programmgesteuerte Löschen des Bildschirms mit **<SHIFT> <CLR/HOME>** wird durch ein revers (negativ) dargestelltes Herz repräsentiert.

```
10 REM SPRINGENDER BALL
20 PRINT "{CLEAR}"
25 FOR X = 1 TO 10 : PRINT "{DOWN}" : NEXT
30 FOR BL = 1 TO 40
40 PRINT" O{LEFT}";: REM O = SHIFT Q
50 FOR TM = 1 TO 5
60 NEXT TM
70 NEXT BL
75 REM BEWEGT DEN BALL NACH LINKS
80 FOR BL = 40 TO 1 STEP -1
90 PRINT " {LEFT}{LEFT}O{LEFT}";
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

Wenn Sie das Programm richtig abgetippt haben, so erscheint nach dem Start ein Ball, der zwischen dem linken und dem rechten Rand hin- und her reflektiert wird.
Auf Seite 44 haben wir das Programm noch einmal abgedruckt und mit Verbindungslinien versehen, die den Ablauf klarer machen sollen.

In Zeile 10 steht hinter dem REM (von REMark = Anmerkung) ein Kommentar, der den Programmtitel enthält; er hat keinerlei Auswirkung auf den Programmablauf. In Zeile 20 wird der Bildschirm gelöscht.
Zeile 25 enthält 10 mal den Befehl Cursor abwärts; hierdurch wird die Ballbewegung in die Bildschirmmitte verlagert. Wenn Sie die Zeile 25 weglassen, wird sich der Ball am oberen Bildschirmrand hin- und herbewegen. Beachten Sie, daß bei einer FOR ... NEXT-Schleife FOR- und

```
        10 REM SPRINGENDER BALL
┌────> 20 PRINT "{CLEAR}"
│       25 FOR X = 1 TO 10 : PRINT "{DOWN}" : NEXT
│  ┌──> 30 FOR BL = 1 TO 40
│  │    40 PRINT" O{LEFT}";: REM O = SHIFT Q
│  │  ┌> 50 FOR TM = 1 TO 5
│  │  └─ 60 NEXT TM
│  └─── 70 NEXT BL
│       75 REM BEWEGT DEN BALL NACH LINKS
│  ┌──> 80 FOR BL = 40 TO 1 STEP -1
│  │    90 PRINT " {LEFT}{LEFT}O{LEFT}";
│  │  ┌> 100 FOR TM = 1 TO 5
│  │  └─ 110 NEXT TM
│  └─── 120 NEXT BL
└────── 130 GOTO 20
```

NEXT-Anweisung in einer Zeile stehen können. Mehrere BASIC-Befehle in einer Zeile müssen jedoch durch Doppelpunkte getrennt werden.
In Zeile 30 befindet sich der erste Teil einer Schleife, die dafür sorgt, daß sich der Ball horizontal (von links nach rechts) über den gesamten, 40 Spalten breiten, Bildschirm bewegt.
In Zeile 40 wird eine Menge Arbeit geleistet. Als erstes wird ein leeres Feld (auch "Space" genannt) gedruckt, dann wird der Ball gedruckt und schließlich wird der Cursor nach links bewegt, um die Vorbedingung für ein erneutes Löschen des Balls zu schaffen.
Die Schleife in den Zeilen 50 und 60 dient nur dazu, die Bewegung des Balls zu verlangsamen. Ohne diese Schleife wäre der Ball kaum sichtbar.
In Zeile 70 befindet sich die zweite Hälfte der Schleife, deren erste Hälfte sich in Zeile 30 befindet. Jedesmal, wenn diese Schleife durchlaufen wird, bewegt sich der Ball um eine Cursorposition auf dem Bildschirm nach rechts. Wie aus den verbindenden Pfeilen im obigen Programmausdruck deutlich hervorgeht, haben wir eine Schleife in einer Schleife (sog. geschachtelte Schleifen) konstruiert.
Wenn Sie diese Methode, Schleifen innerhalb anderer Schleifen laufen zu lassen, anwenden, so müssen Sie darauf achten, daß die **zuerst** geöffnete Schleife als **letzte** geschlossen wird. Überkreuzungen, die Sie z. B. durch Vertauschen der Zeilen 60 und 70 herbeiführen können, führen zu Fehlermeldungen.
In den Zeilen 80 bis 120 wird der umgekehrte Bewegungsablauf auf dem Bildschirm erzeugt. Da die beiden Bewegungsrichtungen auf dem Bildschirm jedoch nicht völlig gleichberechtigt sind, sehen die einander entsprechenden Zeilen 40 und 90 etwas unterschiedlich aus.
Zum Schluß geht das Programm zurück zur Zeile 20 und alles fängt wieder von vorn an.

Zum besseren Verständnis des Programms bringen wir noch eine kleine Änderung in Zeile 40 an:

```
40 PRINT "O";
```

Starten Sie das Programm noch einmal und schauen Sie was passiert. Da wir die Cursorsteuerung weggelassen haben, bleibt der Ball auf dem Bildschirm, bis er durch den "reflektierten" von rechts kommenden Ball gelöscht wird.

INPUT

In den bis jetzt besprochenen Programmen bestand keine Möglichkeit, den Ablauf zu beeinflussen, ohne das Programm selbst zu ändern. Benutzt man die INPUT-Anweisung (INPUT = Eingabe), so hat man die Möglichkeit, während des Programmablaufs Informationen an das Programm zu übergeben.
Das folgende Programm soll Ihnen eine Vorstellung davon geben, wie man mit der INPUT-Anweisung arbeitet:

```
10 INPUT A$
20 PRINT "SIE GABEN EIN: ";A$
30 PRINT
40 GOTO 10
RUN
? COMMODORE 64          Ihre Eingabe
SIE GABEN EIN: COMMODORE 64          Antwort des Computers
```

An der Stelle, wo Sie einen INPUT programmiert haben, hält das Programm an, und ein Fragezeichen signalisiert, daß der Computer eine Eingabe von der Tastatur erwartet. Geben Sie nun irgend eine Folge von Zeichen ein und drücken die **<RETURN>**-Taste, so druckt der Computer aus: SIE GABEN EIN und dahinter den von Ihnen eingegebenen Text.
Um ein Programm komfortabler zu gestalten, kann man zusammen mit dem INPUT einen Kommentar auf den Bildschirm bringen, der z. B. angibt, welche Eingabe erwartet wird. Dieser Kommentar darf allerdings

nicht länger als 38 Zeichen sein. Das Format (d. h. soviel wie Schreibweise) der INPUT-Anweisung ist:

```
INPUT "KOMMENTAR"; Variable
```

Dabei kann die Variable eine Stringvariable oder eine numerische Variable sein.
Wenn Sie auf den Kommentar verzichten wollen, brauchen Sie nur einzugeben: INPUT VARIABLE

Bem.: Um aus diesem Programm "auszusteigen", müssen Sie **<RUN/STOP> <RESTORE>** drücken.

Das folgende Programm, das Temperaturangaben in FAHRENHEIT und CELSIUS ineinander umrechnet, benutzt vieles von dem, was Sie bis jetzt gelernt haben:

```
5 PRINT"{CLEAR}"
10 PRINT"AUSGABE IN FAHRENHEIT ODER CELSIUS (F/C)":INPUT A$
20 IF A$="" THEN 10
30 IF A$="C" THEN 100
40 IF A$="F" THEN 50
45 GOTO 10
50 INPUT"EINGABE VON CELSIUS: ";C
60 F = (C*9)/5+32
70 PRINT C;"GRAD CELSIUS = ";F;"GRAD FAHRENHEIT"
80 PRINT
90 GOTO 10
100 INPUT "EINGABE VON FAHRENHEIT: ";F
110 C = (F-32)*5/9
120 PRINT F;"GRAD FAHRENHEIT = ";C;"GRAD CELSIUS"
130 PRINT
140 GOTO 10
```

Zusammen mit dem INPUT in Zeile 10 wird ein Text auf dem Bildschirm ausgegeben.
In den Zeilen 20, 30 und 40 wird die Eingabe überprüft. Wenn "nichts" eingegeben wird, d. h. nur die **<RETURN>**-Taste gedrückt wurde, so wird das in Zeile 20 festgestellt; das Programm geht nach Zeile 10 zurück und wartet auf eine neue Eingabe. Wurde "C" eingegeben, so bewirkt die Zeile 30 einen Sprung in die Zeile 100, dort beginnt der Programmteil, der Fahrenheit in Celsius umrechnet. Nach Eingabe von "F" springt das Programm nach 50. Jede andere Eingabe führt (wegen Zeile 45) zu einem Sprung nach 10 und damit zu einer erneuten Abfrage.

Wenn das Programm richtig eingetippt wurde, so fragt es ab, welche Umrechnung gewünscht wird, verlangt dann die Eingabe des entsprechenden Temperaturwertes und rechnet um.
Die Umrechnungsformeln können in jedem Grundlehrbuch der Physik nachgelesen werden; ihre Umsetzung in ein BASIC-Programm ist sehr einfach.
Nachdem die Umrechnung erfolgt ist und der entsprechende Wert ausgedruckt wurde, kehrt das Programm wieder in die Zeile 10 zurück und wartet auf ein neue Eingabe. Der Bildschirm könnte zu diesem Zeitpunkt etwa so aussehen.

```
AUSGABE IN FAHRENHEIT ODER CELSIUS (F/C)

? C
EINGABE VON FAHRENHEIT: ? 32
 32 GRAD FAHRENHEIT =  0 GRAD CELSIUS

AUSGABE IN FAHRENHEIT ODER CELSIUS (F/C)

? ■
```

Wenn das Programm zufriedenstellend gelaufen ist, sollten Sie es zur späteren Verwendung auf Band oder Diskette abspeichern.

GET

Mit Hilfe der GET-Anweisung können Sie jeweils ein Zeichen von der Tastatur in den Rechnerspeicher einlesen ohne die **<RETURN>**-Taste zu betätigen. Die Eingabe von Daten läßt sich dadurch in vielen Fällen erheblich beschleunigen. Das eingegebene Zeichen wird der Variablen zugeordnet, die hinter der GET-Anweisung steht. Die folgende Routine illustriert, wie GET arbeitet:

```
PRINT "{CLEAR}"
10 GET A$: IF A$="" THEN 10
20 PRINT A$;
30 GOTO 10
```

Wenn Sie das Programm starten, so wird als erstes der Bildschirm gelöscht. Drücken Sie eine Taste, so bewirkt Zeile 20, daß das zugehörige Zeichen auf dem Bildschirm ausgedruckt wird. Dann wartet die GET-Anweisung auf eine neue Eingabe. Beachten Sie, daß das Zeichen erst durch den PRINT-Befehl auf dem Bildschirm erscheint.
Der zweite Befehl in Zeile 10 ist sehr wichtig. Da GET ununterbrochen arbeitet (im Gegensatz zu INPUT, wo ja die Eingabe durch <RETURN> bestätigt wird), wird geprüft, ob eine Taste gedrückt wurde. Solange das nicht der Fall ist (d. h. solange A$ = ""), bleibt das Programm in Zeile 10. Überprüfen Sie doch mal, was passiert, wenn Sie den zweiten Teil der Zeile weglassen.
Um das Programm anzuhalten müssen Sie **<RUN/STOP> <RESTORE>** drücken. Das Temperatur-Umrechnungs-Programm können Sie leicht mit GET-Anweisungen ausstatten, wenn Sie die Zeilen 10, 20 und 40 in folgender Weise umschreiben und Zeile 45 löschen:

```
10 PRINT"AUSGABE IN FAHRENHEIT ODER CELSIUS (F/C)"
20 GET A$:IF A$="" THEN 20
40 IF A$<>"F" THEN 20
45 :
```

DIE ERZEUGUNG VON ZUFALLSZAHLEN

Ihr COMMODORE 64 enthält eine große Zahl eingebauter Funktionen. Sie können diese Funktionen im Rahmen von BASIC-Programmen wie Unterprogramme benutzen. Dazu brauchen Sie nur die gewünschte Funktion mit ihrem Namen aufzurufen.
Sowohl beim Schreiben von Lernprogrammen wie auch von Spielprogrammen ist es oft von Vorteil, Zufallszahlen zu verwenden; so z.B. wenn Sie ein Würfelspiel simulieren wollen. Für diese Zwecke benützt man sinnvollerweise die RND-Funktion (abgeleitet von RaNDom = zufällig). Tippen Sie das folgende Programm ein, um zu sehen, wie die RND-Funktion arbeitet:

```
10 FOR X = 1 TO 10
20 PRINT RND(1),
30 NEXT
```

Nach Beendigung des Programms sollte der Bildschirm etwa wie folgt aussehen:

```
.125879587          .258874559
.598762589          .145587988
.598526982          .225698774
.369854725          .158798789
.587412365          .265879589
```

Sie haben andere Zahlen? Nun, kein Wunder, die Zahlen sollen ja auch **zufällig** sein.
Lassen Sie das Programm ein paar mal laufen und vergewissern Sie sich, daß stets andere Zahlen ausgedruckt werden. Allen Zahlen gemeinsam ist nur die Eigenschaft, daß sie zwischen 0 und 1 liegen (jedoch nie die Werte 0 und 1 selbst annehmen). Außerdem erhalten wir 9-stellige Dezimalbrüche, oder sogar Zahlen in wissenschaftlicher Notation.
Was machen wir nun, wenn wir ein Würfelspiel darstellen wollen und dafür Zahlen im Bereich von 0 bis 6 und dazu noch ganzzahlig brauchen? Gehen wir erst einmal das erste Problem an, nämlich die Änderung des Bereichs, in dem die Zufallszahlen liegen.
Ändern Sie zu diesem Zweck die Zeile 20 Ihres Programms wie folgt ab:

```
20 PRINT 6*RND(1),

RUN
 4.65860249                    2.50799065
 5.97566542                    4.85598756
 5.08027654                    2.88939395
 2.18540619                    4.62204427
 5.53725942                    3.95162546
```

Nun liegen die Zahlen etwa für ein Würfelspiel im richtigen Bereich, aber die Nachkommastellen stören noch. Aber auch für diesen Fall haben wir eine eingebaute Funktion in unserem COMMODORE 64:

```
20 PRINT INT(6*RND(1)),

RUN
 1        5         5         3
 0        1         4         2
 3        1
```

Die INT-Funktion (abgeleitet von INTeger = ganze Zahl) schneidet die Nachkommastellen einer Dezimalzahl ab.
Ändern Sie die Zeile 20 noch einmal wie folgt ab und starten Sie das Programm:

20 PRINT INT(6*RND(1))

Nun stört nur noch eine Tatsache; statt der Zahlen zwischen 1 und 6 erhalten wir welche zwischen 0 und 5. Wir ändern deshalb die Zeile 20 ein letztes Mal:

20 PRINT INT(6*RND(1))+1

Jetzt liefert uns das Programm die gewünschten Zahlen.
Sie können auf diese Weise Zufallszahlen in jedem gewünschten Bereich der Zahlenskala erzeugen. Die Größe des Bereichs wird vorgegeben durch die Zahl mit der Sie die RND-Funktion multiplizieren. Durch Hinzuaddieren einer Zahl können Sie diesen Bereich beliebig verschieben. Wenn Sie nur ganze Zahlen erhalten wollen, wenden Sie noch die INT-Funktion an. Schematisch ausgedrückt sieht das etwa so aus:

ZAHL = INT(BEREICH*RND(1))+VERSCHIEBUNG

ZAHLENRATESPIEL

Nachdem wir uns so lange mit dem Erklären der Zufallszahlen aufgehalten haben, wollen wir endlich auch eine Anwendung bringen. Das folgende Programm zeigt nicht nur eine sinnvolle Anwendung der RND-Funktion, sondern bringt auch einige neue Programmiertricks.
Die Zufallszahl, die in diesem Programm verwendet wird, wurde der Variablen NM zugeordnet.

```
1 REM ZAHLENRATESPIEL
2 PRINT "{CLEAR}"
5 INPUT "ZAHLEN OBERGRENZE ";LI
10 NM = INT(LI*RND(1))+1
15 CN = 0
20 PRINT "ES KANN LOSGEHEN"
30 INPUT "IHR TIP";GU
35 CN = CN + 1
40 IF GU > NM THEN PRINT "MEINE ZAHL IST KLEINER":PRINT:GOTO 30
50 IF GU < NM THEN PRINT "MEINE ZAHL IST GROESSER":PRINT:GOTO 30
60 IF GU = NM THEN PRINT "TOLL SIE HABEN DIE ZAHL ERRATEN"
65 PRINT "MIT NUR ";CN;"VERSUCHEN.":PRINT
70 PRINT "WOLLEN SIE NOCH EINMAL (J/N)?";
80 GET AN$:IF AN$="" THEN 80
90 IF AN$ = "J" THEN 2
100 IF AN$ <> "N" THEN 80
```

Sie können zu Beginn des Programms festlegen, welche maximale Größe die Zahl, die Sie erraten sollen, annehmen kann. Dann können Sie mit dem Raten beginnen.
Das Programm kommentiert Ihre Eingaben und gibt Ihnen Hinweise darauf, wie groß die zu erratende Zahl ist. Dies wird programmtechnisch

```
ZAHLEN OBERGRENZE ? 25
ES KANN LOSGEHEN
IHR TIP? 15
MEINE ZAHL IST KLEINER

IHR TIP? 10
MEINE ZAHL IST KLEINER

IHR TIP? 5
TOLL SIE HABEN DIE ZAHL ERRATEN
MIT NUR  3 VERSUCHEN.

WOLLEN SIE NOCH EINMAL (J/N)?
```

mit Hilfe von IF ... THEN-Anweisungen realisiert, in denen die vom Computer ausgewählte Zahl mit Ihrer Eingabe verglichen wird.
Versuchen Sie nun einmal das Programm so abzuändern, daß der Benutzer auch die untere Grenze der vom Rechner vorgegebenen Zufallszahlen eingeben kann.
Bei jedem Rateversuch, den Sie machen, wird die Variable CN um 1 erhöht, um die Anzahl der Fehlversuche zu ermitteln, die ja nach Auffinden der "richtigen" Zahl vom Computer zusammen mit dem Text "TOLL SIE HABEN DIE ZAHL ERRATEN" ausgegeben wird. Nun können Sie mit einem neuen Spiel beginnen, in dem der COMMODORE 64 natürlich eine neue Zufallszahl vorgibt.

PROGRAMMIERTIPS:
In den Zeilen 40 und 50 wurden mehrere Befehle durch Doppelpunkte getrennt. Das erspart Ihnen nicht nur überflüssiges Tippen, sondern spart auch Speicherplatz. In denselben Zeilen wurden auch zusätzliche PRINT-Befehle untergebracht, die beim Ausdruck auf dem Bildschirm Leerzeilen erzeugen und dadurch den Bildschirm übersichtlicher gestalten. Da wir von vornherein die Zeilennummern in 10er-Abständen gewählt haben, konnten wir auf einfache Weise nachträglich die Routine zum Zählen der Fehlversuche (Zeilen 15, 35 und 65) einbauen.

WÜRFELSPIEL

Das folgende Programm simuliert ein Würfelspiel mit zwei Würfeln. Viel Glück!

```
5 PRINT "VERSUCHE DEIN GLUECK?"
10 PRINT "ROTER WUERFEL   = ";INT(6*RND(1))+1
20 PRINT "WEISSER WUERFEL = ";INT(6*RND(1))+1
30 PRINT "DRUECKE SPACE FUER WEITERE WUERFE":PRINT
40 GET A$: IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
```

Überprüfen Sie Ihr Wissen über das Programmieren in BASIC und versuchen Sie zu verstehen, wie das Programm aufgebaut ist.

ZUFALLSGRAFIKEN

Daß man auch Grafiken auf Zufallszahlen aufbauen kann, sehen Sie an folgendem kleinen Programm:

```
10 PRINT "{CLEAR}"
20 PRINT CHR$(205.5 + RND(1));
30 GOTO 20
```

Das Wichtigste in diesem Programm steht in der Zeile 20, die die CHR$-Funktion enthält. Je nachdem welche Zahl zwischen 0 und 255 Sie in Klammern hinter den Funktionsnamen setzen, erhalten Sie jeweils ein anderes Zeichen. Die Codetabelle, aus der hervorgeht, wie die Zahlen und Zeichen einander zugeordnet sind, ist im Anhang F dieses Handbuchs abgedruckt.
Sie können den Zahlenwert, der einem Zeichen zugeordnet ist, aber auch selbst ermitteln, wenn Sie die ASC-Funktion benutzen. Sie tippen nur ein:

```
PRINT ASC ("X")
```

Wobei X für das Zeichen steht, dessen Zahlencode Sie wissen wollen. Es kann sich hierbei um jedes druckbare Zeichen, einschließlich der Grafikzeichen, handeln. Der Code, um den es hier geht, wird im allgemeinen als ASCII-Code bezeichnet. Die ASC-Funktion ist die Umkehrfunktion der CHR$-Funktion. D. h., wenn Sie die mit ASC ermittelte Zahl (z. B. Y) in die CHR$-Funktion einsetzen, so erhalten Sie nach Eingabe von

```
PRINT CHR$(Y)
```

das Zeichen X ausgedruckt.
Tippen Sie nun ein

```
PRINT CHR$(205); CHR$(206)
```

so erhalten Sie die Grafikzeichen ausgedruckt, die sich vorn rechts auf der M- und der N-Taste befinden, und aus denen sich ein Irrgarten von der Art aufbauen läßt, wie ihn unser kleines Programm erzeugt hat.

Wenden wir nun die Formel 205.5 + RND(1) an, so werden Zahlen zwischen 205.5 und 206.5 erzeugt. Die Wahrscheinlichkeiten dafür, daß eine Zahl größer oder kleiner als 206 ist, sind gleich groß. Da die CHR$-Funktion die Nachkommastellen ignoriert, werden also die Zeichen mit den Codes 205 und 206 mit gleicher Häufigkeit ausgedruckt.
Sie können diese Wahrscheinlichkeit und damit das Aussehen der Grafik dadurch verändern, daß Sie den Ausgangswert 205.5 um einige Zehntel vergrößern oder vermindern.

KAPITEL 5

FORTGESCHRITTENE GRAFIK- UND FARB-PROGRAMMIERUNG

- Farbe und Grafik
- Farbgebung mit PRINT-Befehlen
- Farb-CHR$-Codes
- PEEK und POKE
- Weitere Ballspiele

FARBE UND GRAFIK

Wir haben nun schon einige der hochentwickelten Programmiermöglichkeiten des COMMODORE 64 kennengelernt. Eine der faszinierendsten Eigenschaften dieses Computers besteht jedoch in der Möglichkeit, Farbgrafiken zu erstellen.
Einfache Beispiele waren das "Springball"-Programm und der Irrgarten. Aber die Möglichkeiten Ihres Computers sind unvergleichlich größer. In diesem Abschnitt werden wir Ihnen zeigen, wie Sie Farbgrafiken erstellen und in eigenen Spielen einsetzen können. Da wir uns bis jetzt auf die Rechenmöglichkeiten unseres Computers konzentriert haben, haben wir uns auf die Standardfarben des Bildschirms (hellblaue Schrift auf dunkelblauem Hintergrund mit hellblauem Rand) beschränkt.
Nun wollen wir Ihnen zeigen, wie Sie diese Farben ändern können und wie Sie die vielen Grafiksymbole Ihres COMMODORE 64 einsetzen können.

FARBGEBUNG MIT PRINT-BEFEHLEN

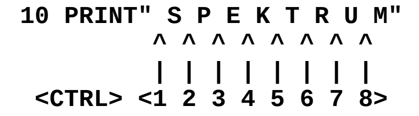
Wenn Sie den Farbjustagetest in Kapitel 1 gemacht haben, so können Sie sich sicher daran erinnern, daß man die Zeichenfarbe einfach dadurch ändern kann, daß man die **<CTRL>**-Taste und eine der Zifferntasten drückt. Aber wie kann man die Farben im Rahmen eines Programms ändern?
Wie wir im "Springball"-Programm gesehen haben, kann man Befehle, die von der Tastatur eingegeben werden können, durch Verwendung zusammen mit PRINT-Befehlen auch im Programm einsetzen.
Sie haben eine Skala von 16 Zeichenfarben zur Verfügung. Durch Benutzung der **<CTRL>**-Taste und der Zifferntasten können Sie folgende erzeugen:

1	2	3	4	5	6	7	8
schwarz	weiß	rot	türkis	violett	grün	blau	gelb

Durch Verwendung der **<C=>**-Taste zusammen mit den Zifferntasten erhalten Sie zusätzlich folgende Farben:

1	2	3	4	5	6	7	8
orange	braun	hellrot	grau 1	grau 2	hellgrün	hellblau	grau 3

Tippen Sie nun NEW ein und machen folgendes Experiment:
Nach dem Zeilenanfang 10 PRINT" drücken Sie zusammen die **<CTRL>**- und die **<1>**-Taste. Lassen Sie dann die Tasten los und drücken die **<S>**-Taste. Nun drücken Sie die **<CTRL>**-Taste zusammen mit der **<2>**-Taste und dann die **<P>**-Taste allein.
Wählen Sie auf diese Weise eine Farbe nach der anderen an und geben dazwischen die Buchstaben des Wortes SPEKTRUM ein.




```
RUN
SPEKTRUM
```

Wie bei der Cursorsteuerung werden auch die Farbsteuerzeichen als graphische Zeichen dargestellt.
Bei gleichzeitigem Drücken von **<CTRL>** und **<3>** (natürlich nur nachdem Sie vorher auf **<">** gedrückt haben) erscheint das Pfundzeichen (£) und **<CTRL> <7>** ergibt den Pfeil nach links (←). Die nachfolgende Tabelle gibt eine Zusammenstellung dieser Farbcodes.

TASTE	FARBE	AUSGABE	TASTE	FARBE	AUSGABE
<CTRL> <1>	SCHWARZ		**<C=> <1>**	ORANGE	
<CTRL> <2>	WEISS		**<C=> <2>**	BRAUN	
<CTRL> <3>	ROT	£	**<C=> <3>**	HELLROT	
<CTRL> <4>	TÜRKIS		**<C=> <4>**	GRAU 1	
<CTRL> <5>	VIOLETT		**<C=> <5>**	GRAU 2	
<CTRL> <6>	GRÜN		**<C=> <6>**	HELLGRÜN	
<CTRL> <7>	BLAU	←	**<C=> <7>**	HELLBLAU	
<CTRL> <8>	GELB		**<C=> <8>**	GRAU 3	

Wie Sie schon bemerkt haben, werden die Steuerzeichen nur beim Listen des Programms sichtbar, bei der Programmausführung, d. h., beim Drucken des Wortes SPEKTRUM tauchen Sie nicht mehr auf, nur der TEXT wird (mit den entsprechenden Farbvariationen) ausgegeben. Spielen Sie nun einige Möglichkeiten durch, um mit der Farbsteuerung vertraut zu werden, und vergessen Sie nicht, daß Sie noch weitere Farbmöglichkeiten im Zusammenhang mit der **<C=>**-Taste haben.

Bem.:
Nach Beendigung eines Programms, in dem Sie die Farbsteuerung angewandt haben, bleibt der Computer in dem Modus, den Sie als letztes eingeschaltet hatten. Sie kommen in den Normalmodus (hellblau, dunkelblau) zurück, wenn Sie **<RUN/STOP> <RESTORE>** drücken.

FARB-CHR$-CODES

Wie Sie dem Anhang F (Liste der CHR$-Codes) entnehmen können, haben die Farben (aber auch z. B. die Cursor-Steuerbefehle) Zahlencodes. Diese Codes können zusammen mit dem PRINT-Befehl direkt benutzt werden, sie haben dieselbe Wirkung wie das Drücken der **<CTRL>** und der entsprechenden Zifferntaste. Probieren Sie z. B. folgendes Beispiel aus:

```
NEW
10 PRINT CHR$(147) : REM CLR/HOME
20 PRINT CHR$(30);"CHR$(30) FAERBT MICH?"
RUN
CHR$(30) FAERBT MICH?
```

Die Schrift sollte nun grün sein. In vielen Fällen ist die Verwendung der CHR$-Funktion zur Farbsteuerung wesentlich einfacher, als die Verwendung der Farbtasten. Das nachfolgende Programm erzeugt farbige Balken auf dem Bildschirm. Beachten Sie, daß sich die Zeilen 40-190 nur in den Zeilennummern und den Argumenten der CHR$-Funktion unterscheiden. Sie brauchen also nur einmal den vollen BASIC-Text einzutippen und können dann durch Ändern der Zeilennummer und der CHR$-Funktion die anderen Zeilen erzeugen (Siehe weiter unten: Edierhinweise).

```
1 REM AUTOMATISCHE FARBBALKEN
5 PRINTCHR$(147) : REM CHR$(147) = CLR/HOME
10 PRINT CHR$(18); "      "; : REM REVERSER BALKEN
20 CL = INT (16*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190
40 PRINTCHR$(5);: GOTO 10
50 PRINTCHR$(28);: GOTO 10
60 PRINTCHR$(30);: GOTO 10
70 PRINTCHR$(31);: GOTO 10
```

```
80 PRINTCHR$(144);: GOTO 10
90 PRINTCHR$(156);: GOTO 10
100 PRINTCHR$(158);: GOTO 10
110 PRINTCHR$(159);: GOTO 10
120 PRINTCHR$(129);: GOTO 10
130 PRINTCHR$(149);: GOTO 10
140 PRINTCHR$(150);: GOTO 10
150 PRINTCHR$(151);: GOTO 10
160 PRINTCHR$(152);: GOTO 10
170 PRINTCHR$(153);: GOTO 10
180 PRINTCHR$(154);: GOTO 10
190 PRINTCHR$(155);: GOTO 10
```

Nach dem Eintippen der Zeilen 1 bis 40 sollte Ihr Bildschirm wie folgt aussehen:

```
1 REM AUTOMATISCHE FARBBALKEN
5 PRINTCHR$(147) : REM CHR$(147) = CLR/HOME
10 PRINT CHR$(18); "       "; : REM REVERSER BALKEN
20 CL = INT (16*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190
40 PRINTCHR$(5);: GOTO 10
■
```

EDIERHINWEISE

Bewegen Sie zuerst den Cursor mit Hilfe der Cursor-Steuertasten auf die "4" der Zeilennummer 40. Tippen Sie nun eine "5", so daß die Zeilennummer 50 entsteht. Steuern Sie nun den Cursor zwischen die Klammern der CHR$-Funktion. Durch Drücken von **<SHIFT> <INST/DEL>** schaffen Sie dort Platz für eine zweistellige Zahl und tippen dann die "28" ein. Nach dem Drücken der **<RETURN>**-Taste sollte der Bildschirm wie folgt aussehen.

```
1 REM AUTOMATISCHE FARBBALKEN
5 PRINTCHR$(147) : REM CHR$(147) = CLR/HOME
10 PRINT CHR$(18); "       "; : REM REVERSER BALKEN
20 CL = INT (16*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190
50 PRINTCHR$(28);: GOTO 10
■
```

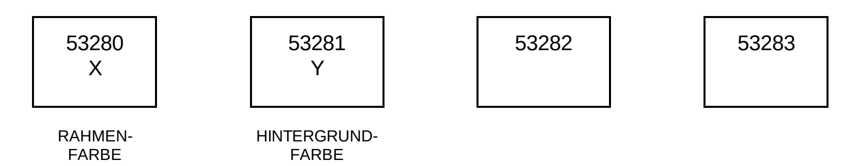
Wenn Sie sich das Programm neu auflisten lassen, werden Sie feststellen, daß die Zeile 40 weiterhin existiert. Wenn Sie jetzt in der neuen Zeile 50 in gleicher Weise Zeilennummer und CHR$-Code verändern, können Sie eine weitere Zeile erzeugen, und auf diese Weise das Programm mit geringem Aufwand fertigstellen. Vergleichen Sie zum Schluß noch einmal den gesamten Programmtext und starten Sie es dann.

Hier eine kurze Beschreibung, wie das Programm funktioniert:
Wahrscheinlich ist Ihnen die Wirkung der meisten Befehle klar, bis auf den Befehl in Zeile 30.
Aber lassen Sie uns trotzdem noch einmal kurz den Programmverlauf verfolgen.
In Zeile 10 wird der CHR$-Code für **<CLR/HOME>** ausgedruckt, was das Löschen des Bildschirms bewirkt.
In Zeile 10 wird der REVERS-Modus eingeschaltet und 6 Leerstellen werden ausgedruckt, die als Balken in der Zeichenfarbe erscheinen. Beim ersten Durchlauf wird als Farbe die Standardfarbe hellblau auftauchen.
Zeile 20 ist unser "Arbeitspferd", hier werden die Zufallszahlen für die Farbwahl erzeugt.
Zeile 30 enthält eine Variation der IF ... THEN Anweisung. Die ON ... GOTO Anweisung läßt das Programm verzweigen, wobei die Wahl einer der hinter GOTO aufgeführten Sprungadressen von der Zahl CL abhängt, die zwischen ON und GOTO steht. Hat CL den Wert 1, so wird die erste Sprungadresse gewählt (in unserem Beispiel Zeile 40), hat CL den Wert 4, so wird die vierte Sprungadresse gewählt etc.
Die Zeilen 40-190 wandeln nun die Zufallszahlen zwischen 1 und 8 in CHR$-Codes für die Farbwahl um und lassen dann das Programm in die Zeile 10 zurückspringen, wo das Spiel von neuem beginnt.

PEEK UND POKE

Nun werden wir eine Methode kennenlernen, wie wir uns im Computer "umschauen" können und Informationen an von uns ausgewählte Stellen bringen, um den Computer zu steuern.
In derselben Weise, wie man sich die Variablen als "Schachteln" vorstellen kann, in die Information "eingefüllt" werden kann, so kann man sich auch vorstellen, daß die Speicherplätze des COMMODORE 64 solchen "Schachteln" entsprechen. Der Inhalt einiger Speicherplätze hat nun ganz bestimmte Bedeutungen. So gibt es Speicher, wo der Computer "nachschaut", welche Bildschirm- oder Rahmenfarbe eingeschaltet ist, welches Zeichen auf dem Bildschirm angezeigt werden soll, welche Farbe es haben soll, wo es stehen soll usw.
Wenn man den Inhalt dieser Speicherplätze ändert, so kann man also auch die oben angeführten Parameter bestimmen. Man kann Farben

ändern, Objekte auf dem Bildschirm erscheinen und sich bewegen lassen und sogar Töne erzeugen und zu Musikstücken zusammensetzen. Schreibt man Werte direkt in den Computerspeicher, so geschieht das mit Hilfe des POKE-Befehls. Man sagt deshalb auch, man "poked" einen Wert in eine Speicherstelle. Man kann Speicherstellen etwa auf folgende Art darstellen



Hier haben wir 4 Speicherplätze dargestellt, von denen zwei die Bildschirm- und Hintergrundfarbe bestimmen.
Tippen Sie bitte folgendes ein:

POKE 53281,7

Nach dem Drücken der **<RETURN>**-Taste erhalten wir einen gelben Bildschirm, da wir den Wert "7" - diese Zahl steht für die Farbe "gelb" -, in die Speicherstelle "gepoked" haben, die die Bildschirmfarbe bestimmt.
Versuchen Sie dasselbe mit anderen Zahlenwerten. Sie können jede Zahl zwischen 0 und 255 verwenden; sinnvoll sind jedoch nur die Zahlen von 0 bis 15.
Der nachstehenden Tabelle können Sie entnehmen, wie die verschiedenen Zahlenwerte den Farben zugeordnet sind.

0	Schwarz	8	orange
1	weiß	9	braun
2	rot	10	hellrot
3	türkis	11	grau 1
4	violett	12	grau 2
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	grau 3

Nun wollen wir uns verschiedene Kombinationen von Hintergrund- und Rahmenfarben anschauen. Dabei hilft uns folgendes Programm:

```
NEW
10 FOR BA = 0 TO 15
20 FOR BO = 0 TO 15
30 POKE 53280,BA
40 POKE 53281,BO
50 FOR X = 1 TO 500 : NEXT X
60 NEXT BO: NEXT BA

RUN
```

Zwei Schleifen werden ineinander geschachtelt, um alle Kombinationen zu erfassen. Die zusätzliche Schleife in Zeile 50 verzögert den ganzen Vorgang nur ein bißchen.
Wenn das Programm abgelaufen ist, tippen Sie ein:

```
? PEEK (53280) AND 15
```

Als Antwort sollte eine "15" auf dem Bildschirm stehen. Dies ist sinnvoll, da als letzter Wert eine "15" in das RAHMEN-Farbregister geschrieben wurde.
Durch die logische Verknüpfung "AND" blenden Sie die Zahlen, die größer als 15 sind, aus. Was das genau bedeutet, läßt sich nur im Rahmen der Arithmetik der binären Zahlen beschreiben. Darauf wird im Kapitel über die "SPRITES" näher eingegangen.
Wenn wir nun die Werte, die in den entsprechenden Registern stehen, während des "Farbwechselprogramms" auf dem Bildschirm ausdrucken lassen wollen, müssen wir die folgende Programmzeile hinzufügen:

```
25 PRINT CHR$(147); "RAHMEN = ";PEEK(53288) AND 15,
"HINTERGRUND = "; PEEK (53281) AND 15
```

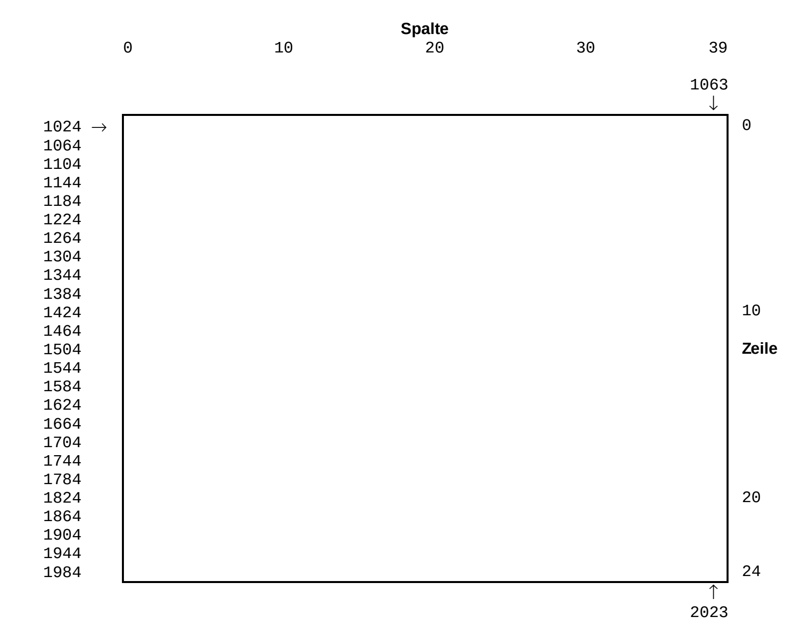
BILDSCHIRMGRAFIK

Bis jetzt haben wir die Zeichen "sequentiell" auf dem Bildschirm gedruckt, d. h. eins nach dem anderen, es sei denn wir hätten eine neue Zeile eingeschaltet, oder mit "," den Ausdruck formatiert.
Den Cursor haben wir bisher durch PRINT-Befehle gesteuert. Damit läßt sich jeder Punkt des Bildschirms erreichen; die Methode ist jedoch im allgemeinen langsam und verbraucht wertvollen Programmspeicherplatz.

Aber wie es Speicherplätze im COMMODORE 64 gibt, die die Bildschirmfarben bestimmen, so gibt es auch Speicherplätze, die den verschiedenen Bildschirmpositionen zugeordnet sind.

DER BILDSCHIRMSPEICHER

Da der Bildschirm unseres Computers aus 25 Zeilen a 40 Zeichen besteht, können wir 1000 Zeichen auf ihm unterbringen und brauchen natürlich genauso viele Speicherplätze, die Informationen darüber enthalten, welche Zeichen sich in den einzelnen Bildschirmpositionen befinden. Wir können uns den Bildschirm als rechtwinkliges Gitter vorstellen (siehe unten), und jeder "Masche" in diesem Gitter entspricht eine Speicherzelle.
Sie können in diese Speicherzellen Werte zwischen 0 und 255 hineinschreiben ("poken"); welche Zeichen Sie dann an der entsprechenden Stelle auf dem Bildschirm ausgedruckt erhalten, können Sie der Tabelle im Anhang E entnehmen.



Der Bildschirmspeicher des COMMODORE 64 beginnt bei 1024 und geht bis zur Speicherstelle 2023. Die Speicherstelle 1024 entspricht dabei der linken oberen Ecke des Bildschirms und die Adressen nehmen

jeweils von links nach rechts zu. Zur weiteren Orientierung betrachten Sie bitte das Schema auf Seite 63.
Nehmen wir nun an, Sie wollen einen Ball auf dem Bildschirm hin- und herspringen lassen. Der Ball soll sich zu Beginn etwa in der Mitte des Bildschirms befinden: Spalte 20, Zeile 12. Die zugehörige Speicheradresse wird dann wie folgt berechnet:

BILDSCHIRMADRESSE = 1024 + SPALTE + 40 * ZEILE

Die Bildschirmadresse unseres Balls hat also den Wert

1024 + 20 + 40 * 12 = 1524

Löschen Sie nun den Bildschirm mit **<SHIFT> <CLR/HOME>** und tippen Sie ein:

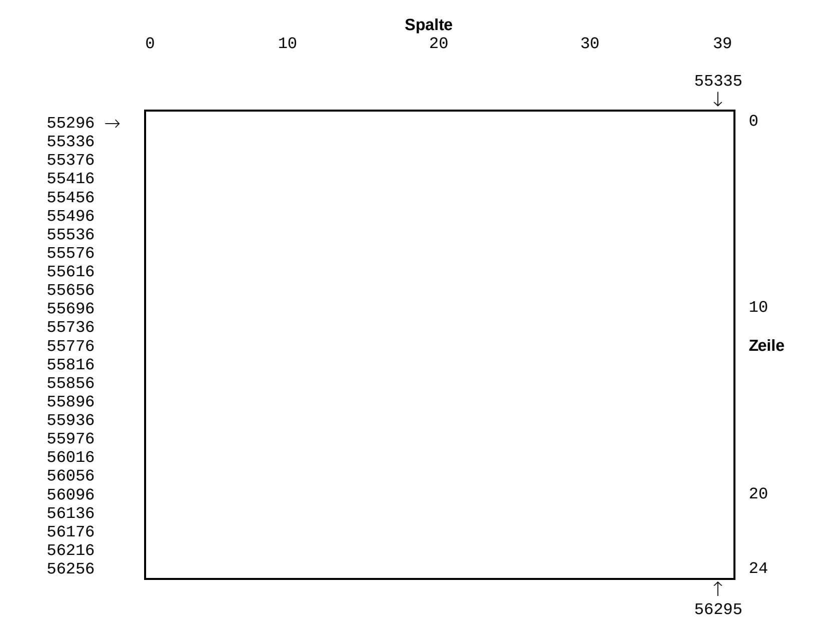
```
POKE 1524,81
      │   └─── ZEICHENCODE
      └─────── BILDSCHIRMADRESSE
```

DER FARBSPEICHER

Sie haben nun einen Ball in der Mitte des Bildschirms erzeugt. Sie haben dies erreicht, ohne einen PRINT-Befehl zu benutzen; Sie haben direkt einen Wert in den Bildschirmspeicher geschrieben. Der Ball der erschien war hellblau. Es gibt jedoch einen Speicher in Ihrem COMMODORE 64, wo Sie durch Ändern der Speicherinhalte die Farben von einzelnen Zeichen auf dem Bildschirm bestimmen können. Tippen Sie nun folgendes ein:

```
POKE 55796,2
       │   └─── FARBE
       └─────── BILDSCHIRMADRESSE
```

Der Ball wird daraufhin rot.
Da Sie außer der Information über das Zeichen in einer bestimmten Bildschirmposition auch eine Farbinformation brauchen, entsprechen jeder Position auf dem Bildschirm zwei Speicherstellen; eine im Bildschirmspeicher und eine im Farbspeicher. Der Farbspeicher beginnt bei 55296 (linke obere Ecke) und hat natürlich, wie der Bildschirmspeicher, 1000 Speicherstellen.



Die Farbcodes liegen zwischen 0 und 15 und entsprechen denen, die wir benutzt haben, um die Farbe von Hintergrund und Rahmen zu ändern (siehe S. 61). Die Formel zur Berechnung der Farbspeicheradresse ist bis auf den Startwert dieselbe wie bei der Bildschirmadresse:

FARBSPEICHERADRESSE = 55296 + SPALTE + 40 * ZEILE

EIN WEITERES SPRINGBALLSPIEL

Auf dieser Seite ist das Programm eines Ballspiels abgedruckt, das eine entscheidende Verbesserung gegenüber dem früheren Beispiel bedeutet. Es arbeitet nicht mehr mit PRINT-, sondern mit POKE-Befehlen. POKE-Anweisungen lassen sich wesentlich flexibler handhaben und erlauben die Darstellung komplizierter Bewegungsvorgänge auf dem Bildschirm.

```
10 PRINT"{CLEAR}" : REM CLR/HOME
20 POKE 53280,7 : POKE 53281,0
30 X=1:Y=1
40 DX=1:DY=1
50 POKE 55296+X+40*Y,1
55 POKE 1024+X+40*Y,81
60 FOR T = 1 TO 10 : NEXT
```

```
70 POKE 1024+X+40*Y,32
80 X = X+DX
90 IF X = 0 OR X = 39 THEN DX = -DX
100 Y = Y+DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
120 GOTO 50
```

In Zeile 10 wird der Bildschirm gelöscht; in Zeile 20 wird als Hintergrundfarbe schwarz und als Rahmenfarbe gelb gewählt.
Die Variablen X und Y in Zeile 30 stehen für die Zeile und die Spalte, in der sich der Ball momentan befindet. Die Variablen DX und DY in Zeile 40 geben die horizontale und vertikale Bewegungsrichtung des Balls an. DX = + 1 entspricht einer Bewegung nach rechts, DX = - 1 einer Bewegung nach links, analog entsprechen DY = + 1 bzw. DY = - 1 Bewegungen nach unten bzw. oben. In den Zeilen 50 und 55 wird der Ball in der durch Zeilen- und Spaltennummer bestimmten Position ausgedruckt und in Zeile 60 taucht wieder die bekannte Verzögerungsschleife auf; schließlich wollen wir den Ball auf dem Bildschirm ja auch sehen.
In Zeile 70 wird der Ball durch Überschreiben mit einem Leerzeichen ("Space") gelöscht.
In Zeile 80 wird durch Addition von DX der Ball in der richtigen Richtung bewegt; das Vorzeichen von DX wird umgedreht, wenn in Zeile 90 festgestellt wird, daß der Ball den linken oder rechten Rand berührt. In den Zeilen 100 und 110 geschieht dasselbe für den oberen und unteren Rand. Die Zeile 120 bewirkt einen Sprung in die Zeile 50, wo der Ball in der neu berechneten Position auf dem Bildschirm ausgegeben wird.
Wenn Sie in Zeile 55 die 81 gegen eine andere Zahl austauschen, so können Sie den Ball durch ein beliebiges Zeichen ersetzen.
Durch folgende Ergänzung können wir das Programm noch ein bißchen intelligenter machen:

```
21 FOR L = 1 TO 10
25 POKE 1024+INT(RND(1)*1000),166
27 NEXT L
85 IF PEEK(1024+X+40*Y)=166 THEN DX= -DX:GOTO 80
105 IF PEEK(1024+X+40*Y)=166 THEN DY= -DY:GOTO 100
```

Die Zeilen 21 bis 27 besetzen zufällig gewählte Bildschirmpositionen mit Hindernissen. In den Zeilen 85 und 105 wird mit Hilfe der PEEK-Funktion geprüft, ob der Ball gegen ein Hindernis stößt; ist dies der Fall, so wird die Bewegungsrichtung geändert.

KAPITEL 6

SPRITE GRAFIK

- **Was sind SPRITES**
- **Die Konstruktion von SPRITES**
- **Binäre Arithmetik**

EINLEITUNG

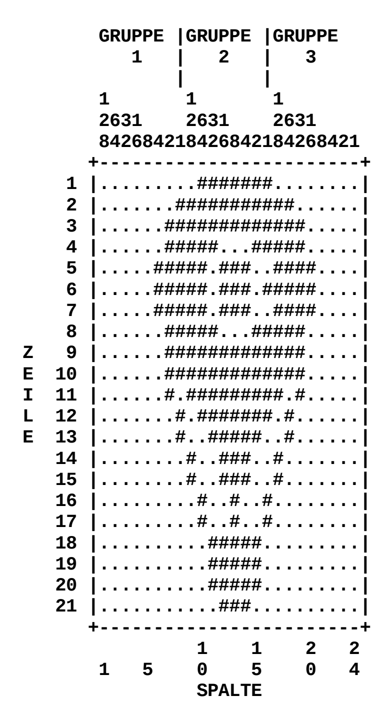
In den vorhergehenden Kapiteln haben wir gesehen, wie wir mit dem PRINT-Befehl den Bildschirm als Tabelle formatieren können und wie wir mit Hilfe des POKE-Befehls an beliebigen Stellen des Bildschirms Zeichen ausdrucken können.
Die Konstruktion von bewegten Darstellungen verursacht mit beiden Methoden einige Schwierigkeiten, da die Objekte aus vorgefertigten grafischen Symbolen zusammengesetzt werden müssen. Weiterhin bringt das Bewegen und Kontrollieren dieser Objekte einen großen Aufwand an Programmierbefehlen mit sich. Durch die begrenzte Zahl von Grafiksymbolen ist man schließlich in der Formgebung der Objekte stark eingeschränkt.
Durch die Verwendung von SPRITES fallen die meisten der aufgeführten Probleme weg. Ein SPRITE stellt ein frei programmiertes Objekt in hochauflösender Grafik dar, dem durch BASIC-Befehle jede beliebige Form gegeben werden kann. Durch einfache Angabe der Position kann das SPRITE auf dem Bildschirm verschoben werden. Die nötigen Berechnungen werden vom COMMODORE 64 automatisch intern erledigt. Aber die SPRITES haben noch mehr Vorteile.
Ihre Farbe kann geändert werden; der Zusammenstoß zweier SPRITES kann auf einfache Weise registriert werden; ein SPRITE kann sich vor oder hinter einem anderen vorbeibewegen und man kann mit einem Befehl seine Größe verändern.
All diesen Vorteilen stehen nur relativ geringe Schwierigkeiten beim Programmieren gegenüber. Zugegeben, Sie müssen noch einiges darüber lernen, wie der COMMODORE 64 arbeitet und wie er Zahlen intern verarbeitet. Aber schließlich ist das doch ganz interessant und auch gar nicht so schwierig; und wenn Sie alle Beispiele sorgfältig durcharbeiten, so werden Sie bald mit selbstkonstruierten SPRITES die verblüffendsten Kunststücke anstellen.

DIE KONSTRUKTION VON SPRITES

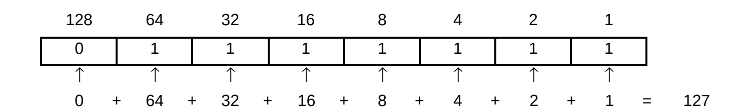
Die Sprites werden von einem speziellen Grafik-Baustein im COMMODORE 64, dem VIC (**V**ideo **I**nterface **C**hip), unterstützt. Die Arbeit, die das Konstruieren der Sprites, das Kontrollieren ihrer Bewegungen und Positionen und die Farbgebung macht, wird zum größten Teil von diesem Chip übernommen.
Der Bereich, in dem die Sprites generiert werden, besteht aus 46 Speicherstellen von der Art, wie wir Sie schon bei der Behandlung des Bildschirm- und Farbspeichers kennengelernt haben. Diese Speicherstellen kann man sich in 8 Zellen, sogenannte Bits, unterteilt denken, die einzeln an- und ausgeschaltet werden können und auf diese Weise die Form der Sprites bestimmen. In welchem Zustand (an oder aus) die einzelnen Bits sind, wird durch die Zahl bestimmt, die in das betreffende Register geschrieben wird - doch davon später mehr.
Zusätzlich zu diesen speziellen Registern werden wir auch den Hauptspeicher des COMMODORE 64 benutzen, um Informationen über die Form der Sprites zu speichern. In 8 Speicherstellen (direkt hinter dem Bildschirmspeicher) werden Daten abgelegt, die den Computer darüber informieren, in welchem Speicherbereich die Daten für die Sprites gespeichert sind.
Wie sind die Sprites nun aufgebaut?
Wie Sie wissen, besteht der Bildschirm aus 25 Zeilen zu je 40 Zeichen. Jede der sich daraus ergebenden 1000 Bildschirmpositionen können Sie durch POKE mit einem Zeichen belegen. Diese Zeichen sind jedoch ihrerseits aus einer 8 x 8-Matrix zusammengesetzt. Beim Konstruieren eines Sprites können Sie nun jeden Punkt dieser Matrix einzeln ansprechen und erhalten dadurch eine Auflösung von 320 x 200 (horizontal x vertikal) Punkten für den gesamten Bildschirm. Ein Objekt, daß Sie auf diese Weise zusammensetzen, darf maximal 24 Punkte breit und 21 Punkte hoch sein.
Als Beispiel haben wir aus diesem 24 x 21-Punkte-Feld einen Ballon konstruiert, der auf Seite 70 abgebildet ist.
Am besten entwerfen Sie das Objekt auf fein gerastertem Papier, auf dem Sie ein 24 Kästchen breites und 21 Kästchen hohes Feld markieren. Zeichnen Sie zuerst die Form so ein, wie Sie Ihnen vorschwebt, und füllen Sie dann die Felder aus, die von den von Ihnen gezogenen Linien geschnitten werden. Nachdem Sie auf diese Weise die Form des Sprites festgelegt haben, müssen Sie diese noch in Daten umwandeln, die der Computer verarbeiten kann. Schreiben Sie zu diesem Zweck (wie in der Abbildung auf Seite 70) an den oberen Rand des 24 x 21-Punkte-Feldes

3 mal hintereinander die Zahlenreihe 128, 64, 32, 16, 8, 4, 2, 1. Die Zeilen des Feldes numerieren Sie von 1 bis 21 durch.

Legen Sie sich nun für jede Zeile drei Wertetabellen an, wie sie auf der Seite 71 dargestellt sind. Tabelle 1 entspricht den ersten 8 Positionen einer Zeile in Ihrer Sprite-Darstellung, und die Tabellen 2 und 3 entsprechen den Positionen 9-16 bzw. 17-24. Über die einzelnen Felder Ihrer Wertetabellen schreiben Sie nun wieder die oben angeführte Zahlenreihe von 128 bis 1 (Sie haben sicher schon gemerkt, daß es sich hier um Potenzen von 2 handelt). Wir ordnen nun in unserer Sprite-Zeichnung jedem ausgefüllten Feld die Zahl 1 und jedem leeren Feld die Zahl 0 zu. Dann legen wir für jede Zeile 3 Tabellen an und schreiben die entsprechenden Werte in die einzelnen Felder.



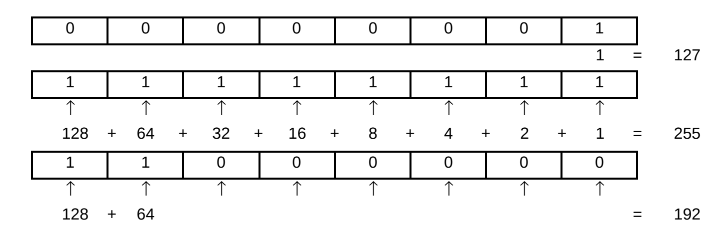
Nehmen wir als Beispiel die Zeile 1 des oben dargestellten Sprites. Die ersten 8 Felder sind leer, d. h. unsere Tabelle 1 enthält lauter Nullen. Um daraus eine Zahl zu errechnen, die der Computer verarbeiten kann, müssen wir den Inhalt jedes Feldes der Tabelle mit dem Wert multiplizieren, der über diesem Feld steht und die Produkte addieren. Da alle Felder 0 enthalten, erhalten wir als erstes Datum (Singular von Daten) als Summe eine 0. Die zweite Tabelle der ersten Reihe haben wir auf der nächsten Seite abgebildet:



Die dritte Tabelle der ersten Reihe enthält wieder lauter Nullen, die Summe ist also auch 0. Die 1. Reihe unseres Sprites wird also durch die Zahlen 0, 127, 0 beschrieben. Um sie leicht in einem Programm verwerten zu können, schreiben wir sie als sogenannte DATA-Zeile:

DATA 0, 127, 0

Als weiteres Beispiel die Daten der zweiten Reihe unseres Sprites:



Für die Reihe zwei erhalten wir also die DATA-Zeile:

DATA 1, 255, 192

Um mit der beschriebenen Methode vertraut zu werden, sollten Sie nun die restlichen 19 DATA-Zeilen selbst berechnen...
Nun haben wir die DATA-Zeilen, aber wie kann man sie einsetzen?
Tippen Sie das folgende Programm ein und starten Sie es:
(Wenn Sie näheres über den DATA- und den dazugehörigen READ-Befehl wissen wollen, schauen Sie bitte in Kapitel 8 nach.)

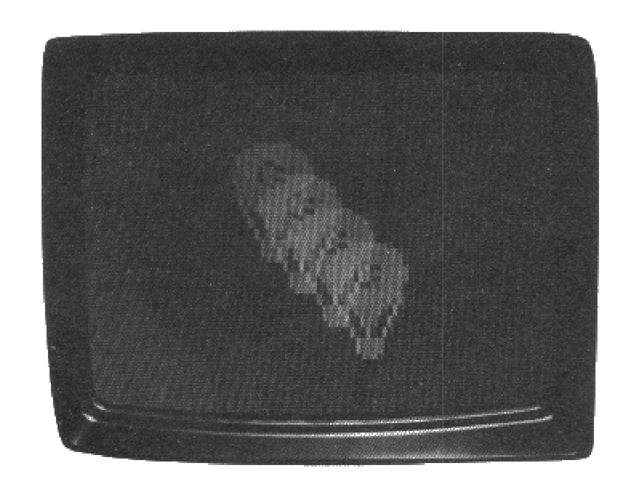
```
1 REM UP, UP, AND AWAY!
5 PRINT "{CLEAR}"
10 V = 53248 : REM BASISADRESSE DES VIC
11 POKE V+21,4 : REM SPRITE 2 AKTIVIEREN
12 POKE 2042,13 : REM DATEN FUER SPRITE 2 AUS BLK 13
20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q : NEXT
                                └── holt die Informationen aus den DATA-Zeilen
30 FOR X = 0 TO 200
40 POKE V+4,X : REM NEUE X-KOORDINATE
50 POKE V+5,X : REM NEUE Y-KOORDINATE
60 NEXT X
70 GOTO 30
```

Hier stehen die Informationen für Q

```
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Wenn Sie alles richtig eingetippt haben, fliegt ein Ballon ruhig über den Himmel.
Um das Programm zu verstehen, müssen Sie wissen, welche Speicherstellen (auch Register genannt) welche Funktionen kontrollieren. Sie können das der folgenden Tabelle entnehmen:

Register	*Beschreibung*
0	X-Koordinate von Sprite 0
1	Y-Koordinate von Sprite 0
2-15	Bedeutung paarweise wie 0 und 1 für die Sprites 1 bis 7
16	Höchstes Bit - X-Koordinate
21	1 = Sprite erscheint; 0 = Sprite verschwindet
29	Sprite wird in X-Richtung vergrößert
23	Sprite wird in Y-Richtung vergrößert
39-46	Farben der Sprites 0-7



Sie müssen außerdem wissen, in welchem 64er Block die Daten eines bestimmten Sprites abgespeichert sind. Diese Daten stehen in den 8 Registern direkt hinter dem Bildschirmspeicher.

	2040	41	42	43	44	45	46	2047
	↑	↑	↑	↑	↑	↑	↑	↑
SPRITE	0	1	2	3	4	5	6	7

Nun wollen wir Schritt für Schritt durchgehen, wie wir unsere Sprites auf den Bildschirm bringen und verschieben können.

1. Poken Sie in die Speicherstelle 21 den "richtigen Wert" (siehe nächste Seite) damit das von Ihnen gewählte Sprite auf dem Bildschirm erscheint.
2. Lassen Sie den Sprite-Pointer (auch "Zeiger") auf die Speicherstelle zeigen, von der ab die Daten des Sprites gelesen werden sollen.
3. Schreiben Sie mit POKE die Daten in diese Speicherstellen.
4. Konstruieren Sie mit einer Schleife die X- und Y-Koordinaten für die Bewegung des Sprites.
5. Sie können zusätzlich die Farben des Sprites oder seine Größe (in X- oder/und Y-Richtung) ändern. Die Parameter für die Größenänderung stehen in den Speicherstellen 29 und 23.

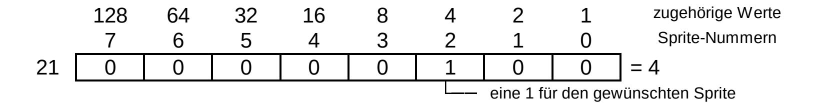
Einige Punkte des Programms könnten trotz der vorangegangenen Erläuterungen noch unklar sein.

Zeile 10:
V = 53248

Die erste Speicheradresse des Video-Chips wird unter V abgespeichert. Zu diesem Wert braucht nur noch die Nummer des Registers addiert zu werden, um die absolute Speicheradresse zu erreichen.

Zeile 11:
POKE V+21,4

Dieser Befehl läßt Sprite Nr. 2 auf dem Bildschirm erscheinen. Der Grund dafür ist, daß im Feld 2 der zugehörigen Tabelle eine 1 steht, wenn in das zugehörige Register eine 4 geschrieben wird.



Jedes Sprite ist im Register 21 repräsentiert, und der Wert 4 im Register entspricht dem Sprite 2. Sprite 3 würde einer 8 entsprechen, beide Sprites zusammen dem Registerinhalt 12 (= 8 + 4). Wenn Sie also die Sprites 2 und 3 "einschalten" wollten, müßten Sie den Befehl POKE V + 21, 12 in Ihr Programm aufnehmen.

Zeile 12:
POKE 2042,13

Der Computer wird angewiesen, die Daten für Sprite Nr. 2 (entspricht Speicherstelle 2042) aus Block 13 des Speichers auszulesen. Ein Sprite "verbraucht" den Inhalt von 63 Speicherstellen oder 63 Bytes. Der Inhalt einer Tabelle, wie wir sie zur Ermittlung der DATA-Zeilen zusammengestellt haben, entspricht einem Byte. Da wir 63 solcher Einheiten brauchen (in jeder der 21 Zeilen 3 Tabellen) müssen wir zur Konstruktion eines Sprites 63 Speicherstellen auslesen. Dies geschieht auf folgende Weise:

```
28 FOR N = 8 TO 62: READ Q: POKE 832+N,Q: NEXT
```

Durch diese Schleife werden 63 Bytes Daten in den 13. Block (1 Block = 64 Bytes) des Speichers eingelesen, der bei der Adresse 832 (= 13*64) beginnt.

```
30 FOR X = 0 TO 200
40 POKE V+4,X ——— X-Koordinate von Sprite 2
50 POKE V+5,X ——— Y-Koordinate von Sprite 2
```

Da die Register 4 und 5 die X- und Y-Koordinaten des Sprites 2 enthalten, bewirkt dieser Programmteil (natürlich nur zusammen mit einem NEXT), daß sich das Sprite 2 diagonal über den Bildschirm bewegt. Da der Koordinatenursprung in der linken oberen Ecke liegt, verläuft die Bewegung von links oben nach rechts unten. Der Computer liest die Daten schnell genug, um die Bewegung kontinuierlich erscheinen zu lassen. Näheres zu diesem Thema steht im Anhang 0.
Wenn sich mehrere Sprites über den Bildschirm bewegen sollen, wird jedem Objekt ein eigener Speicherbereich zugeordnet.
Die Zeile 70 bewirkt einen Rücksprung nach 30, wodurch der ganze Vorgang wiederholt wird. Der Rest des Programms besteht aus DATA-Zeilen, die die Information über die Form des Ballons enthalten.
Fügen Sie nun folgende Zeile zu dem Programm hinzu und starten Sie es erneut:

```
25 POKE V+23,4: POKE V+29,4: REM EXPAND
```

Der Ballon ist jetzt in X- und Y-Richtung doppelt so groß wie vorher. Der Grund dafür ist, daß die Zahl 4 (für Sprite Nr. 2) in die Register 23 und 29 geschrieben wurde. Es ist wichtig zu berücksichtigen, daß die linke obere Ecke des Sprites beim Vergrößerungsvorgang an ihrem Platz bleibt. Um unser Programm noch interessanter zu machen, fügen wir folgende Zeilen hinzu:

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X = 1 TO 190
45 POKE V+6,X
55 POKE V+7,190-X
```

Ein weiteres Sprite (Nummer 3) ist auf dem Bildschirm erschienen, da wir in das Register 21 eine 12 gepoked haben. Die 12 schaltet die 2 und die 3 ein, da 12 = 00001100_2 ist.
Durch die Zeilen 45 und 55 wird Sprite 3 auf dem Bildschirm bewegt. Die Speicherstellen V+6 und V+7 enthalten die X- und Y-Koordinaten des Objekts.
Wenn Sie wollen, daß am Himmel noch ein bißchen mehr los ist, dann bringen Sie an Ihrem Programm die folgenden Ergänzungen an.

```
11 POKE V+21,28
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

Durch den POKE-Befehl in Zeile 11 wurde ein weiteres Sprite auf den Bildschirm gebracht, da in die Position 4 des Registers 21 eine 1 geschrieben wurde. Dieser Position entspricht die Dezimalzahl 16, zusammen mit der 12, die schon im Register stand, ergibt das 28. Nun sind Sprite 2-4 eingeschaltet (00011100_2 = 28).
In Zeile 12 wird festgelegt, daß alle drei Sprites ihre Daten dem 13. Speicherblock entnehmen.
In Zeile 25 werden die Sprites 2 und 3 (daraus ergibt sich die Zahl 12) in X- (V+29) und Y-Richtung (V+23) auf das Doppelte vergrößert.
Zeile 48 bewirkt, daß sich Sprite 3 in Richtung der X-Achse bewegt. Da in Zeile 58 für die Y-Koordinate der konstante Wert 100 vorgegeben wird, bewegt sich Sprite 3 nur horizontal.

NOCH EINIGE BEMERKUNGEN ZU SPRITES

Mit Hilfe derselben Kodierungen (0-15) wie bei den Zeichenfarben (siehe Kapitel 5 oder Anhang G) können Sie die Sprites in 16 verschiedenen Farben darstellen.
Wenn Sie z. B. dem Sprite Nr. 1 die Farbe grün geben wollen, so müssen Sie folgenden Befehl eintippen: POKE V+40,13 (mit V = 53248).
Sie werden vielleicht beim Ausprobieren des Beispielprogramms bemerkt haben, daß die Sprites nie den rechten Bildschirmrand erreichen. Der Grund dafür ist, daß die Bildschirmbreite 320 Punkten entspricht, das X-Register jedoch nur Werte zwischen 0 und 255 enthalten kann. Wie kann man nun ein Objekt dazu bringen, sich über den gesamten Bildschirm zu bewegen?
Man benutzt zu diesem Zweck das Register Nr. 16, über dessen Funktion Sie der Tabelle im Anhang N entnehmen können, daß darin das "höchste Bit", auch MSB = Most Significant Bit genannt, gespeichert ist.
Das ist folgendermaßen zu verstehen: Ist das n-te Bit in diesem Register gesetzt, so befindet sich das n-te Sprite in einer Bildschirmposition, die einem X-Wert > 255 entspricht. Der aktuelle X-Wert ergibt sich dann, wenn man zu dem Wert, der im X-Register steht, 256 hinzuzählt.
Soll z. B. das Sprite Nr. 2 eine X-Position zwischen 256 und 320 einnehmen, so muß im Register Nr. 16 das zweite Bit gesetzt werden. Da 2↑2=4, müssen wir in dieses Register eine 4 poken:

```
POKE V+16,4
```

Nun zählen wir den Inhalt des X-Registers, das zum Sprite Nr. 2 gehört (das ist Register Nr. 4) von 0 bis 63 hoch. Auf diese Weise erreichen Sie die restlichen X-Werte von 256-319.
Sie begreifen das ganze Konzept am besten, wenn Sie das folgende Programm analysieren, das eine leicht abgeänderte Version unseres bisherigen Sprite-Programms darstellt:

```
10 V = 53248 : POKE V+21, 4 : POKE 2042, 13
20 FOR N = 0 TO 62 : READ Q : POKE 832+N, Q : NEXT
25 POKE V+5, 100
30 FOR X = 0 TO 255
40 POKE V+4, X
50 NEXT
60 POKE V+16, 4
70 FOR X = 0 TO 63
```

```
80 POKE V+4, X
90 NEXT
100 POKE V+16, 0
110 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

In Zeile 60 wird das MSB für Sprite Nr. 2 gesetzt. In Zeile 70 beginnt die Schleife, die das Sprite zum rechten Bildschirmrand wandern läßt. Die Zeile 100 ist ebenfalls wichtig, da hier das MSB wieder "abgeschaltet" wird, so daß die Bewegung wieder am linken Bildrand starten kann.

BINÄRARITHMETIK

Eine detaillierte Einführung in die Methode, wie der Computer Zahlen verarbeitet, würde über den Rahmen dieser Einführung hinausgehen; im folgenden sollen jedoch einige Begriffe erläutert werden, die für den Umgang mit Zahlen in Binärdarstellung wichtig sind.

BIT Dies ist die kleinste Informationseinheit, die ein Computer verarbeitet. Ein BIT kann anschaulich als die Information darüber verstanden werden, ob ein Schalter auf "EIN" (entspricht dem Wert 1) oder auf "AUS" (entspricht dem Wert 0) steht. Man sagt auch, ein BIT ist gesetzt (= 1) oder nicht gesetzt (= 0).

Die nächstgrößere "Informationseinheit" ist das BYTE.

BYTE Eine Zusammenstellung von 8 BITS. Je nachdem, ob bestimmte BITS "gesetzt" oder "nicht gesetzt" sind, erhält man 256 verschiedene Kombinationen. Man kann also durch ein BYTE die Zahlen von 0 bis 255 darstellen. Sind in einem BYTE alle BITS **nicht** gesetzt, so kann es auf folgende Weise dargestellt werden:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

Dieses BYTE hat den Wert 0. Sind alle BITS gesetzt, so erhält man:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

Dieses BYTE hat den Wert 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.

REGISTER Ein REGISTER entspricht einer speziellen Adresse im Computer; in ihr kann ein BYTE, d. h. maximal der Wert 255 abgespeichert werden. Durch Änderung der Registerinhalte können Computerfunktionen gesteuert werden. Im Zusammenhang mit den Sprites haben wir z. B. Register kennengelernt, deren Änderung dazu führt, daß Objekte auf dem Bildschirm erscheinen oder ihre Größe in x- bzw. y-Richtung ändern.

DIE UMWANDLUNG VON DER BINÄRDARSTELLUNG IN DIE DEZIMAL-DARSTELLUNG

In der folgenden Tabelle sind die ersten 8 Zweierpotenzen als Binärzahlen dargestellt:

DEZIMALZAHL								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2↑0
0	0	0	0	0	0	1	0	2↑1
0	0	0	0	0	1	0	0	2↑2
0	0	0	0	1	0	0	0	2↑3
0	0	0	1	0	0	0	0	2↑4
0	0	1	0	0	0	0	0	2↑5
0	1	0	0	0	0	0	0	2↑6
1	0	0	0	0	0	0	0	2↑7

Wie bereits erwähnt, können Sie durch ein BYTE alle Zahlen von 0 bis 255 darstellen; da die Speicher unseres Computers ein BYTE enthalten, müssen deswegen die Zahlen, die wir hinein-poken auch zwischen 0 und 255 liegen. Wenn Sie wissen wollen, welcher Dezimalzahl eine bestimmte Binärzahl entspricht, so müssen Sie nur die Zweierpotenzen addieren, an deren Stelle im Bitmuster eine 1 steht. So kommt man zum Wert 255, wenn alle Bits gesetzt sind und erhält z. B. für die Binärzahl "00000011" den Wert 3.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
0	0	0	0	0	0	1	1		
						2	+ 1	=	3

HILFSPROGRAMM: Umwandlung von Binärzahlen in Dezimalzahlen

Das folgende Programm wandelt von Ihnen eingegebene Binärzahlen in Dezimalzahlen um:

```
5 REM BINAER NACH DEZIMAL WANDLUNG
10 INPUT "EINGABE 8-BIT BINAERZAHL :";A$
12 IF LEN(A$) <> 8 THEN PRINT "8 BITS BITTE ..." : GOTO 10
15 TL = 0 : C = 0
20 FOR X = 8 TO 1 STEP -1 : C = C + 1
30 TL = TL + VAL(MID$(A$,C,1))*2↑(X-1)
40 NEXT X
50 PRINT A$; " BINAER = "; TL ;" DEZIMAL"
55 PRINT
60 GOTO 10
```

Die eingegebene Binärzahl wird als String A$ abgespeichert; dieser String wird von rechts nach links mit Hilfe der MID$-Funktion gelesen. Mit der VAL-Funktion wird ermittelt, wie groß der Wert des gelesenen Stringbestandteils ist (0 oder 1) und dieser Wert wird dann mit der entsprechenden Zweierpotenz multipliziert. Die Werte werden aufaddiert und Zeile 50 bewirkt, daß Binär- und Dezimalzahl am Ende des Programms zusammen ausgegeben werden.

KAPITEL 7

MUSIK MIT DEM COMMODORE 64

- **Die Struktur eines Musikprogramms**
- **Beispiel für ein Musikprogramm**
- **Wichtige Klangeinstellungen**
- **Klangeffekte**

Die Tonerzeugung mit dem Computer hat zwei Hauptanwendungsgebiete: das Spielen von Musikstücken und das Erzeugen von Klangeffekten. Wie werden nun kurz darauf eingehen, wie ein Musikprogramm im allgemeinen aufgebaut ist und besprechen dann ein Programm, mit dem Sie experimentieren können.

ZUR STRUKTUR EINES MUSIKPROGRAMMS

Der Klang eines Tones wird durch vier Eigenschaften bestimmt: die **Tonhöhe**, die **Lautstärke**, die **Klangfarbe** und die Art des **Anschlags**. Die beiden letzten Eigenschaften bewirken, daß wir überhaupt verschiedene Instrumente mit dem Gehör unterscheiden können, gerade diese wichtigen Eigenschaften werden Sie daher mit Ihrem Programm auch beeinflussen wollen.
Ihr COMMODORE 64 besitzt zu diesem Zweck wieder einen eigens konstruierten elektronischen Baustein, den **SID** (**S**ound **I**nterface **D**evice). Im SID ist eine Reihe von Speicherplätzen für die Parameter enthalten, die das gewünschte Klangbild zu synthetisieren gestatten. Sie wissen schon, daß Ihr COMMODORE 64 drei Stimmen gleichzeitig erzeugen kann; sehen wir uns zunächst die erste dieser Stimmen an.
Die Basisadresse des SID - wir wollen sie analog zum vorigen Kapitel durch die Variable SI abkürzen - ist 54272:

```
SI = 54272
```

Die **Tonhöhe** wird physikalisch durch die **Frequenz** bestimmt; die Frequenz wird im SID durch einen Parameter gespeichert, der Werte zwischen fast Null und 65000 annehmen kann. Sie haben im vorigen Kapitel gelernt, daß man so große Zahlen nicht in einer Speicherzelle abspeichern kann; wir müssen den Frequenz-Parameter zerlegen in ein höherwertiges und ein niederwertiges Byte, meist als Hi-(High)-Byte und Lo-(Low)-Byte bezeichnet. Diese beiden Bytes belegen die ersten beiden Register im SID:

```
FL = SI      (Frequenz, Lo-Byte)
FH = SI+1    (Frequenz, Hi-Byte)
```

Für die **Lautstärke** sind im SID 16 Stufen vorgesehen, von Null (ausgeschaltet) bis 15 (volle Lautstärke). Der entsprechende Parameter wird in Register 24 abgespeichert:

```
L = SI+24    (Lautstärke)
```

Das war leicht. Nun kommt die **Klangfarbe**: Sie wird im wesentlichen durch die Art der **Wellen** bestimmt, die das betreffende Musikinstrument erzeugt. Der COMMODORE 64 bietet Ihnen vier Grundformen an: **Dreieck**, **Sägezahn**, **Rechteck** und **Rauschen**. Im Programmier-Handbuch werden Sie lernen, wie man diese Grundformen auf raffinierte Weise verändern und durch Filter beeinflussen kann. Hier reichen uns zunächst die Grundformen: Jede von Ihnen wird durch ein Bit im Register 4 kontrolliert:

`W = SI+4` **(Wellenform)**

In dieses Register schreiben Sie zur Auswahl der oben genannten Grundformen einen der Parameter 17, 33, 65 und 129. Wählen Sie 65 (die Rechteck-Welle) müssen Sie zusätzlich noch einen Parameter zwischen Null und 4095 für das sogenannte **Tastverhältnis** (das Verhältnis zwischen "Ein" und "Aus" Ihres Rechtecks) festlegen: die beiden Bytes dieses Parameters kommen in die Register 2 und 3:

`TL = SI+2` **(Tastverhältnis, Lo-Byte)**
`TH = SI+3` **(Tastverhältnis, Hi-Byte)**

Nun noch das Faszinierendste: der **Anschlag**, der **Verlauf** eines Tones. Ihr COMMODORE 64 läßt jeden Ton zunächst auf die im Register 24 eingestellte Lautstärke ansteigen und dann wieder etwas abschwellen, die nun erreichte Lautstärke bleibt erhalten, solange Sie den Ton eingeschaltet lassen; dann klingt der Ton ganz aus. An dieser "**Hüllkurve**" sind, wie Sie sehen, vier Parameter beteiligt, die der SID in zwei weiteren Registern verwaltet:

`A = Sl+5` **(Anschlag)**
`H = Sl+6` **(Halten)**

Jedes dieser Register ist in zwei Teile aufgespalten (vergleichen Sie noch einmal die Einführung in die Binär-Arithmetik im vorigen Kapitel): Der Parameter in den vier höherwertigen Bits von A regelt das **Anklingen** des Tones, der in den vier niederwertigen Bits das **Abschwellen**: kleine Werte bedeuten schnell, hart; große Werte langsam, weich. Dies gilt ebenfalls für die vier niederwertigen Bits von H, die das **Ausklingen** des Tones nach seinem Abschalten kontrollieren. Die höheren vier Bits von H bestimmen die **Lautstärke**, mit der der Ton gehalten wird, der höchste Wert ergibt die in Register 24 vorab eingestellte Lautstärke, kleinere Werte schwächen diese Lautstärke mehr oder weniger ab.

BEISPIELPROGRAMM

Als erstes müssen Sie sich entscheiden, welche **STIMMEN** (oder TONGENERATOREN) Sie benutzen wollen. Für jede dieser Stimmen müssen dann die 4 oben erwähnten Einstellungen (Lautstärke, Wellenform etc.) festgelegt werden. Sie können bis zu drei Stimmen zugleich verwenden, unser Beispiel benutzt jedoch nur die **Stimme Nr. 1**.

`10 SI=54272: FL=SI: FH=SI+1: W=SI+4: A=SI+5: H=SI+6: L=SI+24`	1. Definition der Registeradressen
`20 POKE L,15`	2. Volle Lautstärke
`30 POKE A,16+9`	3. Anschlag
`40 POKE H,4*16+4`	4. Halten und Ausklingen
`50 POKE FH,29:POKE FL,69`	5. Hi und Lo Byte der Frequenz, hier für den Kammerton A. Für andere Töne sind die beiden Werte aus der Tabelle im Anhang P zu entnehmen.
`60 POKE W,17`	6. Wellenform. Muß immer als letztes eingestellt werden, da das niedrigste Bit in diesem Register den Tongenerator ein- bzw. ausschaltet.
`70 FORT=1 TO 500:NEXT`	7. Schleife zur Einstellung der Tondauer.
`80 POKE W,0:POKE A,0: POKE H,0`	8. Ausschalten von Wellenform- und Hüllkurveneinstellungen.

Nach dem Eintippen von RUN können Sie die Note hören, die mit diesem Programm erzeugt wird.

MELODIEN MIT DEM COMMODORE 64

Sie brauchen kein Musiker zu sein, um mit dem COMMODORE 64 Melodien zu erzeugen. Hier ist ein Beispielprogramm, an dem gezeigt wird, wie man das macht. Wir benutzen wieder nur eine Stimme von den dreien, die uns zur Verfügung stehen.
Löschen Sie mit NEW das vorhergehende Programm und tippen Sie dann folgendes ein:

Programmzeile	Erläuterung
`10 REM TONLEITER`	Programmname
`20 SI=54272:FL=SI:FH=SI+1:W=SI+4:A=SI+5:H=SI+6:L=SI+24`	Definition der Registeradressen
`30 POKE L,15`	Volle Lautstärke
`40 POKE A,9`	Anschlag
`50 READ X:READ Y`	Hi-Byte und Lo-Byte der Frequenz aus den Data-Zeilen 130, 140 lesen
`60 IF Y=-1 THEN POKE W,0:END`	Wenn das Programm die - 1 am Ende findet, soll es abschalten
`70 POKE FH,X:POKE FL,Y`	Hi-Byte und Lo-Byte in die Frequenzregister POKEn
`80 POKE W,17`	Wellenform, Generator einschalten
`90 FORT=1TO100:NEXT`	Tondauer
`100 POKE W,0`	Generator ausschalten
`110 FORT=1TO50:NEXT`	Kurze Pause zum Ausklingen
`120 GOTO40`	Nächster Ton
`130 DATA17,103,19,137,21,237,23,59,26,20,29,69,32,219,34,207`	Diese Zahlenpaare stellen die Töne der C-Dur Tonleiter dar, immer abwechselnd ein Hi-Byte, ein Lo-Byte
`140 DATA -1,-1`	Diese (als Frequenz sinnlosen) Daten signalisieren dem Programm in Zeile 60, daß die Tonleiter zu Ende ist.

Wenn wir Töne erzeugen wollen, die denen eines Cembalos ähneln, müssen wir Zeile 80 in folgender Weise abändern.

```
POKE W,33
```

Durch diesen POKE-Befehl wählen wir einen "Sägezahn" als Wellenform; dadurch erhalten wir obertonreichere, "schärfere" Klänge, als es bei der bisher benutzten "Dreieck"-Wellenform der Fall war. Aber die Wahl der Wellenform ist ja nur eine der Möglichkeiten, den Klangcharakter zu bestimmen. Durch eine spezielle Wahl des Anschlag-Wertes können wir aus dem Cembalo ein Banjo machen. Dies geschieht mit dem nachstehenden Poke-Befehl in Zeile 40.

```
POKE A,3
```

Sie können also wie mit einem echten "Synthesizer" den Klang verschiedener Instrumente nachahmen. Wie das gemacht wird, d. h. in welcher Weise wir zu diesem Zweck die betreffenden Registerinhalte ändern müssen, werden wir jetzt besprechen.

WICHTIGE KLANGEINSTELLUNGEN

1. Lautstärke - Die Wahl der Lautstärke gilt für alle 3 Tongeneratoren des COMMODORE 64. Das in Frage kommende Register hat die Adresse 54296; Sie erhalten die maximale Lautstärke, wenn Sie in dieses Register eine 15 poken:

POKE L,15 oder POKE 54296,15

Um die Tongeneratoren auszuschalten, schreiben Sie eine 0 in das Register:

POKE L,O oder POKE 54296,0

Im allgemeinen stellen Sie die Lautstärke zu Beginn eines Musikprogramms fest ein; Sie können jedoch durch programmierte Änderung der Lautstärke interessante Effekte erzielen.

2. WELLENFORM - Wie Sie in unserem Beispiel gesehen haben, bestimmt die Wellenform sehr stark den Klangcharakter eines Tones. Sie können für jede Stimme des COMMODORE 64 die Wellenform getrennt einstellen, dabei haben Sie die Wahl zwischen Dreieck, Sägezahn, Rechteck und Rauschen. Eine Zusammenstellung der entsprechenden Adressen und ihrer Inhalte, die den verschiedenen Stimmen und Wellenformen entsprechen, gibt die nachstehende Tabelle. Wenn Sie z.B. für die 1. Stimme die Wellenform "Dreieck" wählen wollen, müssen Sie folgenden Befehl anwenden:

POKE W,17 oder POKE 54276,17

Die erste Zahl (Adresse) steht also für das Register und die zweite Zahl (Inhalt der Adresse oder des Registers) steht für die jeweilige Wellenform.

DIE EINSTELLUNG DER WELLENFORM

REGISTER				INHALT			
STIMME	1	2	3	RAUSCHEN	RECHTECK	SÄGEZAHN	DREIECK
	4	11	18	129	65	33	17

Wir haben diese Tabelle in Zeile 30 unseres Tonleiterprogramms angewendet. Mit POKE SI+4,17 haben wir das "Dreieck" als Wellenform gewählt, die wir dann zur Änderung des Klangcharakters durch einen "Sägezahn" ersetzt haben, indem wir die 17 in eine 33 umänderten.
Als nächstes wollen wir sehen, wie wir die Hüll-Kurve verändern können, die den Lautstärkeverlauf innerhalb eines Tones bestimmt. Beachten Sie,

daß Sie nur dann einen Ton erhalten, wenn Sie, wie oben beschrieben, auch Lautstärke und Wellenform festlegen.

3. HÜLLKURVEN-EINSTELLUNG - Die Werte für Anschlag und Abschwellen, die wie die Wellenform für jede Stimme getrennt gewählt werden können, werden zusammen durch einen Zahlenwert dargestellt. Während der Anschlag-Parameter die Zeit angibt, in der der Ton bis zur maximalen (vorher eingestellten) Lautstärke ansteigt, ist der Abschwellparameter ein Maß dafür, wie schnell die Lautstärke auf den Halte-Pegel abfällt. Wurde als Halte-Pegel 0 gewählt, so ergibt der Abschwell-Parameter die Abklingzeit bis zur Lautstärke 0 und bestimmt dadurch die Tondauer. Die den einzelnen Stimmen zugeordneten Adressen und die den verschiedenen Anschlag-Einstellungen entsprechenden Werte können der folgenden Tabelle entnommen werden. Die für Anschlag und Abschwellen gewählten Werte werden addiert und die Summe in das entsprechende Register gepoked.

ANSCHLAG-EINSTELLUNGEN

REGISTER				INHALT	
STIMME	1	2	3	ANSCHLAG	ABSCHWELLEN
	5	12	19	15*16 (weich) ... 0*16 (hart)	15 (weich) ... 0 (hart)

Wenn Sie lediglich eine Anschlag-Zeit wählen, z. B. durch POKE 54277,64, so wird die Abschwell-Zeit automatisch 0 gesetzt (und umgekehrt). Durch POKE 54277,66 stellen Sie den Anschlag auf einen mittleren Wert (64=4*16) und das Abschwellen auf einen kleinen Wert (2), der Wert 66 ergibt sich dann als Summe von 64 und 2. Am besten erkennen Sie die Zusammensetzung, wenn Sie statt POKE 54277,66 schreiben: POKE A,4*16+2 (wobei natürlich die Registeradresse A vorher definiert werden muß!)
Wir sind jetzt an einem Punkt angelangt, wo es wahrscheinlich am sinnvollsten ist, das bisher Besprochene im Rahmen eines Programms zusammenzufassen. Tippen Sie NEW ein, drücken Sie die **<RETURN>**-Taste und geben Sie folgendes Programm ein:

```
10 REM EXPERIMENTIERPROGRAMM
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:
   TH=SI+3:W=SI+4: A=SI+5: H=SI+6:
   L= SI+24
30 PRINT"DRÜCKE EINE TASTE!"          Bildschirm-Botschaft
40 GETZ$:IFZ$=""THEN40                Taste gedrückt?
```

```
 50 POKE L,15                    Lautstärke
 60 POKE A,1*16+5                Anschlag und Abschwellen
 70 POKE H,0*16+0                Halten und Ausklingen
 80 POKE TH,8:POKE TL,0          Tastverhältnis
 90 POKE FH,14:POKE FL,162       Frequenz
100 POKE W,17                    Wellenform, Generator einschalten
110 FORT=1TO200:NEXT             Tondauer
120 POKE W,0                     Generator abschalten
130 GOTO40                       Alles von vorn
```

Wir benutzen die Stimme 1 zur Erzeugung eines Tones mit kurzer Anstiegszeit und kurzer Abfallphase nach Erreichen der Maximallautstärke (Zeile 60!). Was dabei herauskommt, klingt etwa wie ein Ball, der in einer Blechtonne hin- und herspringt. Um einen anderen Klang zu erzeugen, ändern wir diese Zeile. Dazu stoppen wir das Programm mit **<RUN/STOP>**, lassen uns das Programm mit LIST (mit nachfolgendem Drücken der **<RETURN>**-Taste) auslasten und ändern die Zeile 60, wie folgt:

```
60 POKE A,11*16+14
```

Wenn wir jetzt **<RETURN>** drücken, übernimmt der Computer die geänderte Zeile in seinen Programmspeicher.
Der Ton, den wir mit dieser Einstellung erhalten, hat etwa den Klang einer Oboe oder eines sonstigen Holzblasinstruments. Experimentieren Sie nun, und ändern Wellenform und Hüllkurve um ein Gefühl dafür zu bekommen, wie die verschiedenen Werte dieser Parameter den Toncharakter verändern.
Mit der Halte-Einstellung können Sie festlegen, welche Lautstärke der Ton nach dem Anschlagen beibehält. Die Dauer des Tons wird dabei wie gewöhnlich mit einer FOR ... NEXT-Schleife geregelt. Ähnlich wie beim vorigen Register werden Halten und Ausklingen des Tons durch einen Zahlenwert festgelegt, der sich durch Addition aus den Werten ermitteln läßt, die in der folgenden Tabelle aufgeführt sind:

HALTEN / AUSKLINGEN

REGISTER				INHALT	
STIMME	1	2	3	HALTEN	AUSKLINGEN
	6	13	20	15*16 (laut) ... 0*16 (stumm)	15 (langsam) ... 0 (schnell)

Ersetzen Sie die Nullen in Zeile 70 durch irgendwelche Werte bis maximal 15 und hören Sie, was dabei herauskommt!

5. DIE WAHL DER STIMMEN UND DER NOTEN - Wie Sie bereits erfahren haben, müssen Sie zur Erzeugung eines Tones zwei Werte eingeben, die wir Hi-Byte und Lo-Byte der Frequenz genannt haben. Die Zuordnung dieser Werte zu den Notennamen können Sie der Tabelle in Anhang P entnehmen.
Da den Stimmen unterschiedliche Adressen zugeordnet sind (siehe folgende Tabelle), können Sie die drei Stimmen Ihres COMMODORE 64 unabhängig voneinander programmieren und auf diese Weise z.B. dreistimmige Musikstücke erstellen.

ADRESSEN DER DREI TONGENERATOREN UND POKE-WERTE HI-BYTE UND LO-BYTE DER TÖNE DER MITTLEREN (5.) OKTAVE

REGISTER				INHALTE FÜR NOTEN DER 5. OKTAVE												
STIMME	1	2	3	C	C#	D	D#	E	F	F#	G	G#	A	A#	H	C
HI-BYTE	1	8	15	35	37	39	41	44	46	49	52	55	58	62	66	70
LO-BYTE	0	7	14	3	24	77	163	29	188	132	117	148	226	98	24	6

Um den Ton C mit der Stimme 1 zu erzeugen, müssen Sie folgende POKE-Befehle verwenden:

```
        POKE 54273,35:POKE 54272,3
oder    POKE Sl+1,35:POKE SI,3
```

Denselben Ton mit der Stimme 2 erhalten Sie durch:

```
        POKE 54280,35:POKE 54279,3
oder    POKE Sl+8,35:POKE Sl+7,3
```

WIR PROGRAMMIEREN EIN LIED AUF DEM COMMODORE 64

Mit dem folgenden Beispielprogramm kann man Lieder "komponieren" und wiedergeben; der Computer benutzt dazu die Stimme 1. Beachten Sie bitte, daß in der Programmzeile 110 die Adressen der häufig verwendeten Register numerischen Variablen zugeordnet werden und dadurch im Programm bequemer angewendet werden können. Wenn z.B. die Wellenform gewählt werden soll, so genügt es, im entsprechenden POKE-Befehl den Buchstaben W statt der Zahl 54276 einzusetzen.
Weiterhin sollten Sie sich für die Verwendung in eigenen Programmen merken, wie mit den DATA-Zeilen gearbeitet wird. Im vorliegenden

Programm werden in den DATAS die drei Zahlen, die zur Beschreibung eines Tones notwendig sind, hintereinander abgespeichert. Es handelt sich hierbei um Hi-Byte und Lo-Byte der Frequenz und die TONDAUER.
Die Tondauer wird durch eine Schleife bestimmt, die von 1 bis zu einem bestimmten Wert läuft. Dieser Wert steht in den DATAS jeweils an dritter Stelle. Dabei entspricht 125 einer Achtelnote, 250 einer Viertelnote, 375 einem punktierten Viertel, 500 einer halben und 1000 einer ganzen Note. Je nach Tempoangabe oder musikalischem Geschmack können diese Werte natürlich nach oben oder unten abgeändert werden.
Schauen wir uns nun mal die Zeile 110 an; die 17 und die 103 sind Hi-Byte und Lo-Byte für die Note "C" und die Zahl 250 an der dritten Stelle Stelle bewirkt, daß es eine Viertelnote wird. Auch die zweite Note ist ein Viertel, diesmal ist es aber ein "E" etc.... Sie können in die DATA-Zeilen aber auch selbstgewählte Werte eintragen, die Sie mit Hilfe der Notentabelle im Anhang P ermitteln können. Dabei können Sie Ihre Melodie so lang wählen, wie Speicherplatz in Ihrem COMMODORE 64 verfügbar ist, Sie müssen nur dafür sorgen, daß die letzte Programmzeile den Ausdruck DATA-1,-1,-1 enthält. Zeile 130 sorgt dann dafür, daß das Programm endet, wenn es in dieser Zeile angelangt ist.

```
10 REM MICHAEL ROW THE BOAT ASHORE
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKEL,15:POKETH,13:POKETL,15:POKEA,3*16+15:POKEH,9
40 READ X:READY:READD
50 IFX=-1THENEND
60 POKEFH,X:POKEFL,Y
70 POKEW,65
80 FOR T=1 TO D:NEXT
90 POKEW,0
100 GOTO40
110 DATA17,103,250,21,237,250,26,20,400,21,237,100,26,20,250,29,69,250
120 DATA26,20,250,0,0,250,21,237,250,26,20,250,29,69,1000,26,20,250,0,0,250
130 DATA-1,-1,0
```

KLANGEFFEKTE

Im Unterschied zur Musik sollen Klangeffekte Ereignisse, die auf dem Bildschirm stattfinden, untermalen (Explosion eines Raumschiffs etc.) oder sie sollen den Benutzer eines Programms informieren oder warnen (z. B., daß er gerade im Begriff ist seine Datendiskette zu löschen etc.)
Hier einige Vorschläge, die zum Experimentieren anregen sollen:

1. Ändern Sie die Lautstärke während der Ton erklingt, Sie können damit z. B. einen "Echoeffekt" erzielen.
2. "Springen" Sie schnell zwischen zwei Tonhöhen hin und her, um ein "Tremolo" zu erzielen.

3. Probieren Sie verschiedene Wellenformen aus.
4. Beschäftigen Sie sich eingehend mit der Hüll-Kurve.
5. Durch unterschiedliches Programmieren der drei Stimmen (z. B. den Ton in einer Stimme etwas länger anhalten als in der anderen) kann man überraschende Effekte erzielen.
6. Benutzen Sie die Rechteckwelle und ändern Sie die Pulsbreite (Anhang O)
7. Experimentieren Sie mit dem Rauschgenerator zur Erzeugung von Explosionsgeräuschen, Gewehrfeuer, Schritten etc.
8. Ändern Sie in schneller Folge die Frequenz über mehrere Oktaven hinweg.

KLANGEFFEKTE ZUR DEMONSTRATION

Die hier besprochenen Programmbeispiele können Sie in direkter, oder in abgeänderter Form in eigene BASIC-Programme einbauen. Sie sollen Sie zum Experimentieren anregen und Ihnen zeigen, welche Klangmöglichkeiten in Ihrem COMMODORE 64 stecken.

```
10 REM PUPPE
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKEL,15:POKETH,15:POKETL,15:POKEA,0*16+0:POKEH,15*16
40 POKEW,65
50 FOR X=250TO0STEP-2:POKEFH,40:POKEFL,X:NEXT
60 FOR X=150TO0STEP-4:POKEFH,40:POKEFL,X:NEXT
70 POKEW,0
```

```
10 REM GEWEHRSCHUSS
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 FORX=15TO0STEP-1
40 POKEL,X:POKEA,15:POKEH,0:POKEFH,40:POKEFL,200:POKEW,129
50 NEXT
60 POKEW,0:POKEA,0
```

```
10 REM MOTOREN
20 SI=54272
30 FORK=0TO24:READX:POKESI+K,X:NEXT
40 DATA  9,2,0,3,0,0,240
50 DATA 12,2,0,4,0,0,192
60 DATA 16,2,0,6,0,0,64
70 DATA 0,30,243,31:REM FILTER
80 POKESI+4,65:POKESI+11,65:POKESI+18,65
```

KAPITEL 8

FORTGESCHRITTENES PROGRAMMIEREN

- READ und DATA
- Mittelwert
- Indizierte Variable;
  Eindimensionale Felder
- Dimensionierung
- Würfelspiel
- Zweidimensionale Felder

READ UND DATA

Sie haben zwei Methoden kennengelernt, mit denen man einen Zahlenwert einer Variablen zuordnen kann; durch eine direkte Zuweisung im Programm (z. B. A=2) oder mit einer INPUT-Anweisung. Bei manchen Anwendungen, insbesondere wenn es sich um die Verarbeitung größerer Zahlenmengen handelt, sind jedoch beide Methoden nicht praktikabel.
In solchen Fällen ist es sinnvoll, wie im folgenden Programm vorzugehen:

```
10 READ X
20 PRINT "X IST NUN :"; X
30 GOTO 10
40 DATA 1,34,10.5,16,234.56
RUN
X IST NUN : 1
X IST NUN : 34
X IST NUN : 10.5
X IST NUN : 16
X IST NUN : 234.56

?OUT OF DATA  ERROR IN 10
READY.
■
```

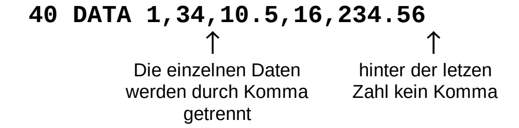
Das Programm ist so aufgebaut, daß die Zeilen 10 bis 30 wiederholt durchlaufen werden. Jedesmal, wenn die Zeile 10 erreicht wird, liest die READ-Anweisung die nächste Zahl in den DATA-Anweisungen. Bei jedem Durchlauf der Schleife wird ein ZEIGER (oder POINTER) um eine Stelle nach hinten versetzt; auf diese Weise "merkt" sich der Computer, wie weit er mit dem Lesen gekommen ist.

```
                POINTER
                  ↓
40 DATA 1,34,10.5,16,234.56
```

Wenn der Computer alle in den DATA-Anweisungen stehenden Zahlen gelesen hat und beim nächsten Leseversuch keine Daten findet, so bricht er das Programm mit der Fehlermeldung OUT OF DATA ERROR ab.

Folgendes Format muß bei den DATA-Anweisungen eingehalten werden:

```
40 DATA 1,34,10.5,16,234.56
```



Die DATA-Anweisungen können ganze Zahlen, Dezimalbrüche oder Zahlen in wissenschaftlicher Notation enthalten, jedoch keine Variablen oder arithmetischen Operationen. Folgende DATA-Zeile ist unzulässig:

```
40 DATA A,23/56,2*5
```

Auch Strings können in DATA-Anweisungen gespeichert werden; Sie müssen dann allerdings, wie im folgenden Beispiel gezeigt ist, auch die zugehörige READ-Anweisung mit einer String-Variablen versehen:

```
10 FOR X = 1 TO 3
15 READ A$
20 PRINT "A$ IST NUN : "; A$
30 NEXT
40 DATA DIES, IST, LUSTIG
RUN
A$ IST NUN : DIES
A$ IST NUN : IST
A$ IST NUN : LUSTIG

READY.
■
```

Im Gegensatz zum letzten Beispiel haben wir die READ-Anweisung in eine FOR ... NEXT-Schleife eingeschlossen, die so oft durchlaufen wird, wie es der Anzahl der Elemente in der DATA-Anweisung entspricht. Wird diese Zahl geändert, so muß auch die Anzahl der Schleifendurchläufe geändert werden. Es ist deswegen in vielen Fällen bequemer, das Ende der Daten durch eine Marke (auch "flag" genannt) zu kennzeichnnen. Diese Marke sollte durch ein Element (Zahl oder String) dargestellt werden, das in Ihren Daten nicht vorkommen kann (z. B. eine negative Zahl, wenn in Ihren DATA-Zeilen das Lebensalter von Personen aufgelistet ist). Mit dem Auftreten dieser Marke kann dann eine Bedingung verknüpft werden, die zum nächsten Programmteil verzweigen läßt.

Wenn Sie die in DATA-Anweisungen gespeicherten Werte in einem Programm mehrfach verwenden wollen, so müssen Sie den Zeiger wieder auf das erste Element zurücksetzen. Dies geschieht mit Hilfe des RESTORE-Befehls. Ergänzen Sie das Programm auf Seite 95 durch die Zeile

```
50 GOTO 10
```

Sie werden die Fehlermeldung OUT OF DATA ERROR erhalten, da nach dem ersten Durchlauf des Programms der DATA-Zeiger auf den letzten String zeigt und nach dem Rücksprung nach Zeile 10 keine Daten mehr findet. Sie müssen deshalb noch folgende Zeile hinzufügen:

```
45 RESTORE
```

Das Programm läuft nun beliebig lang, da nach jedem Durchlauf der DATA-Zeiger zurückgestellt wird.

MITTELWERTE

Das folgende Programm stellt eine praktische Anwendung von READ und DATA dar. Zahlen werden aus DATA-Zeilen eingelesen und der Mittelwert wird berechnet.

```
5 T=0 : CT=0
10 READ X
20 IF X=-1 THEN 50 : REM TEST AUF FLAG
25 CT=CT+1
30 T=T+X : REM BERECHNUNG DER SUMME
40 GOTO 10
50 PRINT "ES WURDEN ";CT;"WERTE GELESEN"
60 PRINT "SUMME =";T
70 PRINT "MITTELWERT =";T/CT
80 DATA 75, 80,62,91,87,93,78, -1

RUN
ES WURDEN  7 WERTE GELESEN
SUMME = 566
MITTELWERT = 80.8571429
```

In Zeile 5 werden Schleifenzähler CT und die Zahlensumme T gleich 0 gesetzt. In Zeile 10 werden die Daten gelesen und in Zeile 20 werden sie überprüft ob es sich dabei um eine Marke ("flag") handelt; in unserem Fall wird die Marke durch eine "- 1" dargestellt. Wenn es sich um gültige Daten handelt, so werden sie aufaddiert (Zeile 30) und der Zähler wird um 1 erhöht (Zeile 25).
Handelt es sich beim gelesenen Datum um eine Endmarkierung, so verzweigt das Programm zur Zeile 50. Dort wird ausgegeben, aus wie vielen Werten der Mittelwert berechnet wird. Zeile 60 führt zur Ausgabe der Summe, und in Zeile 70 wird der Mittelwert berechnet und auf dem Bildschirm ausgedruckt. Wenn Sie das Ende Ihrer DATA-Anweisungen mit einer Marke kennzeichnen, können Sie die Anzahl der Elemente beliebig verändern und brauchen sich keine Gedanken über ihre Anzahl zu machen.
Eine Variation in der Anwendung von READ und DATA besteht darin, daß Sie Elemente aus einer DATA-Anweisung verschiedenen Variablen zuordnen. Sie können in solchen Fällen sogar Zahlen und Strings in den DATAS miteinander mischen, wie es in dem folgenden Programm geschehen ist.

```
10 READ N$,A,B,C
20 PRINT N$;"'S PUNKTE SIND :";A;" ";B;" ";C
30 PRINT "UND DER DURCHSCHNITT IST :";(A+B+C)/3
40 PRINT : GOTO 10
50 DATA MIKE, 190, 185, 165, DICK, 225, 245,190
60 DATA JOHN, 155, 185, 205, PAUL, 160, 179,187

RUN
MIKE'S PUNKTE SIND : 190   185   165
UND DER DURCHSCHNITT IST : 180

DICK'S PUNKTE SIND : 225   245   190
UND DER DURCHSCHNITT IST : 220

usw.
```

Sie müssen in solchen Fällen natürlich darauf achten; daß den Stringvariablen bzw. numerischen Variablen auch Strings bzw. Zahlen in den DATA-Anweisungen entsprechen. So erwartet die READ-Anweisung in unserem Programm jeweils einen String und darauffolgend drei Zahlen; dies entspricht auch der Struktur der Daten in den DATA-Zeilen.

INDIZIERTE VARIABLEN

Wir haben bis jetzt nur einfache Variablen der Form A, XY, K1, C$ etc. benutzt. Der Variablenname bestand immer aus einem Buchstaben, oder aus der Kombination eines Buchstabens mit einem zweiten, bzw. mit einer Ziffer.
Die Anzahl der Variablennamen, die Sie auf diese Weise erzeugen können, dürfte in den meisten Fällen ausreichen, aber die Handhabung solcher Variablen in einem Programm ist manchmal recht umständlich.
Wir wollen Sie deshalb in das Konzept der indizierten Variablen einführen:



Eine indizierte Variable besteht also aus einem Buchstaben gefolgt von einer in Klammern gesetzten Zahl, dem Index. Gelesen wird diese Zusammenstellung "A von eins". Passen Sie auf, daß Sie A(1) nicht in einen Topf mit A1 werfen; A1 ist **keine** indizierte Variable.
Indizierte Variablen bezeichnen, wie die "normalen" Variablen auch, Speicherplätze im Computer, die Sie sich etwa wie numerierte Schachteln vorstellen können.

A(0)	0
A(1)	0
A(2)	0
A(3)	0
A(4)	0

Wenn Sie jetzt definieren:

```
A(0)=25 : A(3)=55 : A(4)=-45.3
```

sieht Ihr Speicher so aus:

A(0)	25
A(1)	0
A(2)	0
A(3)	55
A(4)	-45.3

Bem.: Indizierte numerische Variable enthalten den Wert 0, wenn ihnen kein anderer Wert zugewiesen wurde. Es gibt jedoch auch die Möglichkeit, Stringvariablen zu indizieren (z. B. B$(2)); solche Variablen enthalten, wenn ihnen kein String zugewiesen wurde "nichts".

Eine Solche Gruppe indizierter Variablen nennt man ein FELD (oder auch TABELLE). Bis jetzt haben Sie nur die eindimensionalen Felder (mit einem Index) kennengelernt. Später werden wir den mehrdimensionalen Fall behandeln. Indices können komplex zusammengesetzt sein, d. h. sie können z. B. Variablen oder Rechenoperationen enthalten. Erlaubt sind z.B. die folgenden Ausdrücke für indizierte Variablen:

A(X)   A(X+1)   A(2+1)   A(5*3)

Die Ausdrücke in den Klammern werden nach den normalen Rechenregeln für arithmetische Operationen (Kapitel 2) ausgeführt.
Die Grundbegriffe sind Ihnen nun wahrscheinlich klar. Aber was fängt man mit den indizierten Variablen in einem Programm eigentlich an?
Eine Anwendung besteht z. B. darin, Zahlen, die mit READ oder INPUT eingelesen werden, in einer Liste zu speichern. Im nächsten Programm verwenden wir die indizierten Variablen, um einen Mittelwert auf etwas abweichende Art (von der auf S. 94) auszurechnen:

```
5 PRINTCHR$(147)
10 INPUT "WIEVIELE ZAHLEN :";X
20 FOR A=1 TO X
30 PRINT "WERT #";A;:INPUT B(A)
40 NEXT
50 SU=0
60 FOR A=1 TO X
70 SU=SU+B(A)
80 NEXT
90 PRINT : PRINT "MITTELWERT = "; SU/X

RUN

WIEVIELE ZAHLEN :? 5
WERT # 1 ? 125
WERT # 2 ? 167
WERT # 3 ? 189
WERT # 4 ? 167
WERT # 5 ? 158

MITTELWERT =  161.2
```

Man hätte dieses Programmierproblem sicher eleganter lösen können, aber hier soll ja illustriert werden, wie die indizierten Variablen angewendet werden können. In Zeile 10 wird gefragt, wieviele Zahlen eingegeben werden sollen. Die zugeordnete Variable X gibt an, wie oft die Schleife durchlaufen werden muß, in der die zu verarbeitenden Werte eingegeben und der indizierten Variablen B(A) zugeordnet werden. Der Wert von A wird bei jedem Schleifendurchlauf um 1 erhöht, und auf diese Weise werden fortlaufend die indizierten Variablen B(1), B(2), B(3)... etc. erzeugt und den eingegebenen Zahlen zugeordnet. Das hat aber zur Folge, daß wir auf diese Zahlen sehr leicht zugreifen können, um sie auszudrucken oder in irgend einer Weise verarbeiten zu lassen. In den Zeilen 50 bis 80 wird z. B. der Mittelwert errechnet. Sie können sich die Zahlen aber auch auf einfache Weise auf dem Bildschirm ausdrucken lassen. Tippen Sie dazu ein:

FOR A = 1 TO 5 : ? B(A), : NEXT (**<RETURN>**-Taste drücken)

Sie erhalten dann die Inhalte der indizierten Variablen auf dem Bildschirm ausgedruckt.

DIMENSIONIERUNG

Wenn Sie im obigen Beispiel versucht haben, mehr als 10 Werte einzugeben, so haben Sie die Fehlermeldung BAD SUBSCRIPT ERROR IN 30 erhalten. Sie können Felder bis zu 11 Elementen (das entspricht den Indices 0 bis 10 bei eindimensionalen Feldern) wie normale Variablen benutzen; Felder mit mehr als 11 Elementen müssen jedoch "dimensioniert" werden.
Fügen Sie zu diesem Zweck die folgende Zeile zu obigem Programm hinzu:

5 DIM B(100)

Hierdurch teilen Sie Ihrem Computer mit, daß Sie maximal 100 Werte eingeben wollen. Sie können aber auch mit einer Variablen dimensionieren. Löschen Sie dazu erst die Zeile 5 (durch Eintippen von 5 und Drücken der **<RETURN>** -Taste) und geben dann folgende Zeile ein:

15 DIM B(X)

Dadurch wird das Feld exakt so dimensioniert, wie es der Anzahl der Zahlen, die Sie eingeben wollen, entspricht.
Aber bedenken Sie: eine einmal erfolgte Dimensionierung kann im weiteren Verlauf des Programms nicht rückgängig gemacht werden.

Mehrere Dimensionierungen können in folgender Weise zusammengefaßt werden:

```
10 DIM C(29),D(58),E(48)
```

SIMULATION EINES WÜRFELSPIELS

Komplizierte Programme können durch Verwendung von indizierten Variablen knapp und einfach geschrieben werden.
Im nächsten Programm wird eine indizierte Variable verwendet, um zu registrieren, wie oft eine bestimmte Zahl gewürfelt wird.

```
1 REM WUERFEL SIMULATION
2 PRINT CHR$(147)
10 INPUT "WIEVIELE WUERFE ";X
20 FOR L=1 TO X
30 R=INT(6*RND(1))+1
40 F(R)=F(R)+1
50 NEXT L
60 PRINT"WURF","ANZAHL DER WUERFE"
70 FOR C=1 TO 6 : PRINTC,F(C) : NEXT
```

Ein Element des Feldes F(1) ... F(6) wird jeweils um 1 vergrößert, wenn die entsprechende Zahl gewürfelt wird; so wird z. B. bei 2 Augen der Wert von F(2) um 1 erhöht.
In Zeile 10 wird gefragt, wie oft gewürfelt werden soll; die gewünschte Anzahl wird dann in der Variablen X gespeichert.
Eine von Zeile 20 bis Zeile 50 reichende Schleife, in der die Würfe simuliert und die Ergebnisse registriert werden, wird dann X-mal durchlaufen. In Zeile 60 und 70 werden die Ergebnisse in Form einer Tabelle ausgedruckt.
Nach dem Ablauf des Programms kann der Bildschirm etwa wie folgt aussehen:

```
WIEVIELE WUERFE ? 1000
WURF      ANZAHL DER WUERFE
 1         157
 2         153
 3         173
 4         188
 5         152
 6         177
```

Untenstehend haben wir ein Programm abgedruckt, welches das gleiche leistet; jedoch ohne indizierte Variablen zu verwenden. Sie brauchen es nicht abzutippen, aber beachten Sie, um wieviel länger es gegenüber dem vorhergehenden Programm ausgefallen ist.

```
10 INPUT "WIEVIELE WUERFE ";X
20 FOR L=1 TO X
30 R=INT(6*RND(1))+1
40 IFR=1THENF1=F1+1:NEXT
41 IFR=2THENF2=F2+1:NEXT
42 IFR=3THENF3=F3+1:NEXT
43 IFR=4THENF4=F4+1:NEXT
44 IFR=5THENF5=F5+1:NEXT
45 IFR=6THENF6=F6+1:NEXT
60 PRINT"WURF","ANZAHL DER WUERFE"
70 PRINT 1,F1
71 PRINT 2,F2
72 PRINT 3,F3
73 PRINT 4,F4
74 PRINT 5,F5
75 PRINT 6,F6
```

Die Länge des Programms hat sich von 8 auf 16 Zeilen verdoppelt. Bei größeren Programmen, insbesondere bei einer größeren Zahl von Variablen, kann man noch mehr Zeilen und damit Speicherplatz einsparen.

ZWEIDIMENSIONALE FELDER

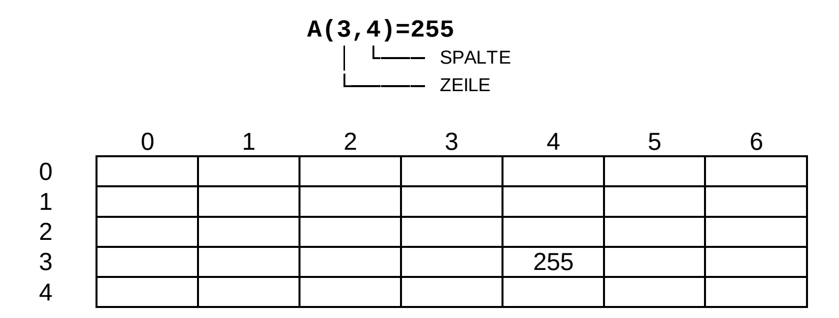
Das Element eines zweidimensionalen Feldes hat folgende Form:

```
A(4,6)
|  └─┴── INDICES
└─────── FELDNAME
```

Anschaulich kann man sich ein solches Feld als ein zweidimensionales Gitter vorstellen:

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

Hierbei geben die Indices an, in welcher Spalte und welcher Zeile dieser Tabelle das betreffende Element steht.



	0	1	2	3	4	5	6
0							
1							
2							
3					255		
4							

Wir können uns also vorstellen, daß die 255 in der dritten Zeile und der vierten Spalte unserer Tabelle steht.
Wie die eindimensionalen, müssen auch die mehrdimensionalen Felder dimensioniert werden. Ein Feld von 21 mal 21 Elementen wird z. B. durch DIM A(20,20) dimensioniert.
Was ist nun eine typische Anwendung für ein zweidimensionales Feld? Nehmen wir einmal an, Sie müßten einen Fragebogen entwerfen, der 4 Fragen enthält, auf die je 3 verschiedene Antworten möglich sind. Ihr Fragebogen würde also etwa wie folgt aufgebaut sein:

FRAGE 1: Sind Sie mit Ihrem Chef zufrieden?

1 = JA 2 = NEIN 3 = UNENTSCHIEDEN

... usw.

Die Auswertung eines solchen Fragebogens könnte in Form der folgenden Tabelle geschehen, die wir uns als zweidimensionales Feld denken.

	ANTWORTEN		
	JA	NEIN	UNENTSCHIEDEN
FRAGE 1			
FRAGE 2			
FRAGE 3			
FRAGE 4			

Das zugehörige Auswerteprogramm ist auf Seite 104 abgedruckt.
Viele Programmiertechniken, die wir bis jetzt besprochen haben, sind in diesem Programm zur Anwendung gekommen. Auch wenn Sie das Programm nicht benutzen wollen, sollten Sie versuchen, seinen Aufbau zu verstehen.

Das Kernstück des Programms ist ein Feld aus 4 mal 3 Elementen. In diesen Elementen werden die Antworten gezählt. Aus Gründen der Übersichtlichkeit haben wir dabei die Elemente A(0,0) bis A(0,4) nicht benutzt.
Im Klartext: Wenn auf die Frage 1 die Antwort "JA" gegeben wurde, so wird A(1,1) um 1 vergrößert - Zeile 1, da es sich um Frage 1 handelt und Spalte 1, da die Antwort "JA" lautet. Eine Beantwortung der Frage 3 mit NEIN würde das Element A(3,2) um 1 vergrößern etc.

```
20 PRINT"{CLEAR}"
30 FOR R=1TO4
40 PRINT"FRAGE NUMMER :";R
50 PRINT"1-JA 2-NEIN 3-ENTHALTUNG"
60 PRINT"WAS IST DIE ANTWORT:";
61 GET C : IFC<1 OR C>3 THEN 61
65 PRINT C : PRINT
70 A(R,C)=A(R,C)+1
80 NEXT R
85 PRINT
90 PRINT"WOLLEN SIE EINE NEUE ANTWORT EINGEBEN":PRINT"ANTWORT (J/N)"
100 GET A$ : IF A$=""THEN 100
110 IF A$="J"THEN20
120 IF A$<>"N"THEN100
130 PRINT"{CLEAR}DIE GESAMTEN ANTWORTEN SIND:":PRINT
140 PRINTSPC(18);"ANTWORT"
141 PRINT"FRAGE","JA","NEIN    ENTHALTUNG"
142 PRINT"-----     ----------------------------"
150 FOR R=1 TO 4
160 PRINT R,A(R,1),A(R,2),A(R,3)
170 NEXT R

RUN

FRAGE NUMMER : 1
1-JA 2-NEIN 3-ENTHALTUNG
WAS IST DIE ANTWORT: 2

FRAGE NUMMER : 2
1-JA 2-NEIN 3-ENTHALTUNG
WAS IST DIE ANTWORT: 1

FRAGE NUMMER : 3
1-JA 2-NEIN 3-ENTHALTUNG
WAS IST DIE ANTWORT: 1


DIE GESAMTEN ANTWORTEN SIND:

                  ANTWORT
FRAGE     JA        NEIN    ENTHALTUNG
-----     ----------------------------
1          7         1         0
2          6         2         0
3          8         0         0
4          2         5         1
```

ANHANG

EINLEITUNG

Nachdem Sie nun mit Ihrem COMMODORE 64 enger vertraut sind, sollen Sie wissen, daß unsere Unterstützung hier nicht endet. Sie werden es vielleicht nicht wissen, aber COMMODORE ist bereits seit 23 Jahren im Geschäft. In den 70er Jahren stellten wir den ersten eigenständigen Personalcomputer vor, PERSONAL ELECTRONIC TRANSACTOR. Seit damals sind wir in vielen Ländern der Welt eine der führenden Computerfirmen geworden. Durch unsere Fähigkeit, Computerchips selbst zu entwickeln und herstellen, sind wir in der Lage, Ihnen fortschrittliche Personalcomputer anbieten zu können, zu einem weit niedrigeren Preis, als man es bei diesem technischen Niveau erwarten könnte.
COMMODORE fühlt sich verpflichtet, nicht nur Sie als Endverbraucher zu unterstützen, sondern auch ein größeres Händlernetz zu pflegen und Zeitschriften, die sich mit neuen Anwendungen und Techniken befassen, wie auch Softwarehäuser, die Anwendungsprogramme auf Kassetten und Disketten entwickeln, zu fördern. Wir publizieren zum Beispiel in mehreren Ländern Zeitschriften, die Programmiertips, Informationen über neue Produkte und Ideen für neue Computeranwendungen enthalten.
Die folgenden ANHÄNGE enthalten Zeichnungen, Tabellen und andere Informationen, die Ihnen helfen sollen, Ihren COMMODORE 64 schneller und effizienter programmieren zu können. Sie enthalten auch wichtige Informationen über die weite Palette interessanter Commodore-Produkte und eine Bibliographie, die Ihnen helfen kann, Ihr Programmiergeschick zu entwickeln.

COMMODORE 64 ZUBEHÖR UND SOFTWARE

ZUBEHÖR

Der COMMODORE 64 bedient VC-20 Speichereinheiten und übriges Zubehör: Datasette, Diskettenlaufwerk, Drucker. Ihr System kann also jederzeit den sich wandelnden Erfordernissen angepaßt werden.

- Datasette Recorder – Diese preiswerte Bandeinheit speichert Programme und Daten auf handelsüblichen Musicassetten. Auch vorgefertigte Programme sind auf diesen Kassetten erhältlich.

- Disk – Das Einfachlaufwerk VC 1541 benutzt 5'/4 Zoll Standarddisketten zum Speichern von Programmen und Daten. Die Diskstation erlaubt schnelleren Datenzugriff. Sie speichert auf jeder Diskette bis zu 170000 Zeichen. Diskstationen sind "intelligent", das heißt, sie besitzen ihren eigenen Mikroprozessor und Speicher. Sie belegen deshalb keinen Speicherplatz im Computer.

- Drucker – Der Drucker VC-1525 erzeugt gedruckte Kopien von Programmen, Daten und Graphiken. Dieser Matrixdrucker verarbeitet 30 Zeichen pro Sekunde und benötigt gewöhnliches Papier für Traktorführungen und andere wenig teure Verbrauchsmaterialien.

- Interface Steckkarten – Eine Anzahl spezieller Steckkarten wird für den COMMODORE 64 zur Verfügung stehen, so daß unterschiedliche Standardgeräte wie Drucker, Modems, Controller, Meßgeräte und andere Instrumente angeschlossen werden können.

Mit einer speziellen IEEE-488 Steckkarte kann der COMMODORE 64 sämtliche CBM Peripherie-Geräte benutzen, einschließlich der Disks und Drucker.
Zusätzlich wird eine Z80-Karte den Betrieb unter CP/M* ermöglichen. Sie erhalten damit Zugang zur größten Quelle von Mikrocomputer-anwendungen.

* CP/M ist ein registriertes Warenzeichen der Digital Research Inc.

SOFTWARE

Für den COMMODORE 64 werden verschiedene Kategorien von Software angeboten. Sie können dann aus einer großen Palette von Programmen aus Bereichen wie Haushalt, persönliche Datenverwaltung, Unterhaltung und Ausbildung auswählen.

Hilfen im Beruf

- Ein elektronisches "Datenblatt" erlaubt es Ihnen, Budgetpläne aufzustellen und zu analysieren, "Was wäre wenn?" Mit einem zugehörigen Grafikprogramm können aus solchen "Datenblättern" anschauliche Grafiken erzeugt werden.
- Finanzielle Vorausplanungen, wie Amortisation von Investitionen werden mit einem Finanzplanungspaket leicht gehandhabt.
- Eine Reihe professioneller Zeit-Management-Programme wird ihnen helfen, Ihre Termine und Arbeitsbelastungen zu optimieren.
- Leicht benutzbare Programme für Datenbanken werden wichtige Informationen auf dem Laufenden halten – Versandkarteien, Telefonlisten, Inventarverzeichnisse,... – und alle Informationen in der für Sie nützlichen Form organisieren.
- Professionelle Textbearbeitungsprogramme können Ihren COMMODORE 64 in einen vollwertigen Schreibplatz verwandeln. Erstellen und Korrigieren von Memoranden, Briefen und anderem Textmaterial wird Ihnen wie im Fluge von der Hand gehen.

UNTERHALTUNG

- Die hochwertigsten Spiele werden in Einsteckmodulen erhältlich sein. Sie nutzen die Möglichkeiten der hochauflösenden Grafik des COMMODORE 64 und seine volle Fähigkeit Töne, Geräusche und Musik zu erzeugen.

AUSBILDUNG

- Der COMMODORE 64 ist ein Tutor, der nie müde wird und Ihnen immer seine volle Aufmerksamkeit schenkt. Neben den vielen CBM-Ausbildungsprogrammen wird der COMMODORE 64 über zusätzliche Sprachen des Erziehungsbereichs wie etwa PILOT, LOGO und weitere fortgeschrittene Pakete verfügen.

FORTGESCHRITTENE KASSETTEN OPERATION

Neben Kopien Ihrer Programme speichert der COMMODORE 64 auf Magnetband die Werte und Variablen und andere Arten von Daten in einer Anordnung, die FILE genannt wird. Sie können so mehr Information speichern, als der Computer auf einmal in seinem Hauptspeicher unterbringen kann.

Befehle, die für Datenfiles benutzt werden, sind OPEN, CLOSE, PRINT#, INPUT# und GET#. Die Systemvariable ST (status) wird benutzt, um z.B. Markierungen auf dem Magnetband zu prüfen.

Um Daten auf Magnetband aufzuzeichnen, wird dasselbe Konzept benutzt, wie bei der Ausgabe auf dem Bildschirm. Nur wird die Information auf den Kassettenrecorder umgelenkt. Dies wird auch erkenntlich an der dafür benutzten Variation des PRINT-Befehls, nämlich PRINT#. Das folgende Programm illustriert die Arbeitsweise:

```
10 PRINT "SCHREIBEN AUF BAND"
20 OPEN 1,1,1,"DATEN FILE"
30 PRINT "TIPPEN SIE IHRE DATEN EIN
   ODER TIPPEN SIE DAS WORT STOP"
40 PRINT
50 INPUT "DATA";A$
60 PRINT#1,A$
70 IF A$ <> "STOP" THEN 40
80 PRINT
90 PRINT "FILE WIRD GESCHLOSSEN"
100 CLOSE 1
```

Das erste, was Sie tun müssen, ist ein File öffnen (in diesem Falle in Zeile 20 unter dem Namen DATEN FILE).

Das Programm fragt in Zeile 50 nach den Daten, die Sie speichern wollen. Zeile 60 schreibt, was Sie eingetippt haben (in der Variablen A$ abgelegt), auf Band. Danach wird erneut nach Daten gefragt.

Tippen Sie das Wort STOP ein, so wird das File geschlossen.

Um wieder an die Daten heranzukommen, spulen Sie das Band zurück und probieren einmal das:

```
10 PRINT "READ-TAPE-PROGRAM"
20 OPEN 1,1,0,"DATEN FILE"
30 PRINT "FILE OPEN"
40 PRINT
50 INPUT#1,A$
60 PRINT A$
70 IF A$ = "STOP" THEN CLOSE 1 : END
80 GOTO 40
```

Wieder muß das "DATEN FILE" geöffnet werden. In Zeile 50 wird A$ gelesen (INPUT#) und in Zeile 60 auf den Bildschirm geschrieben (PRINT). Der ganze Prozeß wird wiederholt, bis "STOP" gelesen wurde, wodurch das Programm beendet wird.
Eine Variation von GET, nämlich GET#, kann ebenfalls benutzt werden, um Daten wieder vom Band herunter zu lesen. Ersetzen Sie die Zeilen 50-80 in dem Programm oben durch:

```
50 GET#1,A$
60 IF A$ = "" THEN CLOSE 1 : END
70 PRINT A$,ASC(A$)
80 GOTO 50
```

ANHANG C

COMMODORE 64 BASIC

Dieses Handbuch hat Ihnen bis jetzt eine Einführung in die Programmiersprache BASIC gegeben – genug um Ihnen ein Gefühl für die Programmierung von Computern und das dabei benutzte Vokabular zu geben. In diesem Anhang finden Sie nun eine vollständige Liste aller Regeln (der Syntax) des COMMODORE 64 BASIC, zusammen mit kurzen Beschreibungen. Probieren Sie die Kommandos einfach aus. Sie können Ihrem Computer keinen dauerhaften Schaden zufügen, wenn Sie Programme eintippen; und Programmieren lernt man immer noch am leichtesten, indem man es tut.
Entsprechend den verschiedenen Arten von Operationen in BASIC ist dieser Anhang in mehrere Sektionen eingeteilt:

1. Variablen und Operatoren: beschreibt die verschiedenen Variablentypen, gültige Variablennamen, arithmetische und logische Operatoren.
2. Kommandos: beschreibt die Kommandos, die gebraucht werden, um Programme zu edieren, sie zu laden, zu speichern und zu löschen.
3. Befehle: beschreibt die BASIC-Befehle, die in numerierten Programmzeilen vorkommen.
4. Funktionen: beschreibt die String-, numerischen und Druckfunktionen.

VARIABLEN

Der COMMODORE 64 benutzt drei Arten von Variablen: Fließkommazahlen (real), ganze Zahlen (integer) und Strings (alphanumerische Zeichenketten).
Variablennamen können aus einem einzelnen Buchstaben, einem Buchstaben mit nachfolgender Ziffer oder zwei Buchstaben bestehen. Integervariablen werden durch ein dem Namen nachgestelltes Prozentzeichen (%) gekennzeichnet. Stringvariablen tragen nach ihrem Namen ein Dollarzeichen ($).

Beispiele:

Namen von Realvariablen: A, A5, BZ
Namen von Integervariablen: A%, A5%, BZ%
Namen von Stringvariablen: A$, A5$, BZ$

Arrays sind Mengen von Variablen, die denselben Namen benutzen und durch Angabe einer besonderen Nummer (Index) als bestimmtes Element dieses Arrays ausgezeichnet werden. Arrays werden durch das DIM Statement definiert. (Statement ist das in der Computerei häufig benutzte englische Wort für Befehl, Aussage). Arrays dürfen von jedem der drei Typen Real, Integer oder String sein. Dem Variablennamen folgt in runden Klammern die Angabe, wieviele Elemente die Liste enthält und wie diese angeordnet sein sollen (wieviele Dimensionen das Array hat).

A(7), BZ%(11), A$(50), PT(20,20)

Bemerkung: Es gibt drei Variablennamen, die der COMMODORE 64 selbst benutzt und die Sie nicht definieren dürfen: ST, TI und TI$.
ST ist die Statusvariable und erhält ihren Wert entsprechend einer zuvor ausgeführten Ein- oder Ausgabe-Operation. ST ändert seinen Wert z.B., wenn ein Problem beim Schreiben oder Lesen von Band und Diskette auftritt.
TI wird von der internen Uhr jede 1/60 Sekunde hochgezählt. Es hat beim Einschalten des Computers den Wert 0 und kann nur zurückgesetzt werden, wenn TI$ geändert wird.
TI$ ist ein String aus sechs Ziffern, der ständig vom System auf den gültigen Stand gebracht wird. Die beiden ersten Ziffern geben die Stunden an, die beiden mittleren Ziffern die Minuten und die beiden letzten Ziffern die Sekunden. Dieser Variablen kann ein beliebiger Wert zugewiesen werden, der von da an ständig erhöht wird.

TI$ = "101530" setzt die Uhr auf 10:15 Uhr und 30 Sekunden.

Die Uhr wird beim Ausschalten des Computers gelöscht und beginnt beim Einschalten mit 0.

OPERATOREN

Die arithmetischen Operatoren werden von folgenden Zeichen gebildet:

+ Addition
– Subtraktion

* Multiplikation
/ Division
↑ Exponentiation (hoch)

Werden in einer Zeile mehrere arithmetische Operatoren verwendet, dann gilt für die Reihenfolge der Berechnung die in der Mathematik übliche "Punkt vor Strich"-Regel. Das heißt, zuerst wird die Exponentiation ausgeführt, dann werden Multiplikationen und Divisionen vor der Addition und Subtraktion berechnet.
Die Reihenfolge der Rechnung kann durch runde Klammern geändert werden, da Operationen in Klammern vor allen anderen ausgeführt werden. Auch das ist das in der Mathematik übliche Verfahren.
Für Vergleiche werden die folgenden Operatoren verwendet:

= gleich
< kleiner als
> größer als
<= kleiner oder gleich
>= größer oder gleich
<> ungleich

Schließlich gibt es noch die logischen Operatoren:

AND UND (zwei Bedingungen treffen gleichzeitig zu)
OR ODER (mindestens eine von zwei Bedingungen trifft zu)
NOT NICHT (die Bedingung trifft nicht zu)

Diese werden gewöhnlich gebraucht um Mehrfachbedingungen in IF ... THEN Statements unterbringen. Zum Beispiel:

IF A = B AND C = D THEN 100 (beide Vergleiche müssen stimmen)
IF A = B OR C = D THEN 100 (mindestens einer der beiden Vergleiche muß stimmen)

KOMMANDOS

CONT (Continue)

Dieses Kommando wird benutzt, um ein Programm fortzusetzen, das durch die STOP-Taste, den Befehl STOP oder den Befehl END im Programm angehalten wurde. Das Programm arbeitet genau an der Stelle weiter, wo es unterbrochen wurde.
CONT geht nicht, wenn Sie vorher Zeilen geändert oder hinzugefügt

haben. CONT geht auch nicht,wenn die Programmunterbrechung durch einen Fehler ausgelöst wurde oder Sie zuvor einen Fehler verursachten. In diesen Fällen erhalten Sie die Meldung: CAN'T CONTINUE ERROR.

LIST

Mit List lassen Sie sich Programmzeilen anzeigen. Sie können sich das ganze Programm oder auch nur bestimmte Zeilen daraus ausgeben lassen.

LIST	zeigt das ganze Programm
LIST 10 –	zeigt das Programm ab Zeile 10 bis zum Ende
LIST 10	zeigt nur Zeile 10
LIST – 10	zeigt das Programm vom Anfang bis Zeile 10
LIST 10 – 20	zeigt die Zeilen 10 bis 20, einschließlich

LOAD

Diesen Befehl benutzen Sie, um ein Programm von Band oder Diskette in den Rechner zu übertragen. Wenn Sie nur LOAD eintippen und die RETURN-Taste drücken, wird das nächste Programm von der Kassette gelesen. Das Kommando kann durch einen Programmnamen in Anführungszeichen ergänzt werden. Dem Namen kann ein Komma und eine Zahl oder numerische Variable folgen, die als Gerätenummer interpretiert wird und die angibt, von welchem Gerät das Programm zu laden ist.
Wird keine Gerätenummer angegeben, nimmt der COMMODORE 64 Gerät #1 an, die Kassetteneinheit. Das andere, gewöhnlich mit dem LOAD-Kommando angegebene Gerät ist das Diskettenlaufwerk, Gerät #8.

LOAD	liest das nächste Programm von der Kassette
LOAD "HALLO"	sucht auf der Kassette das Programm HALLO und lädt es, wenn es gefunden wurde.
LOAD A$	sucht das Programm, dessen Name in der Variablen A$ abgelegt ist.
LOAD "HALLO",8	sucht das Programm mit dem Namen HALLO auf der Diskette.
LOAD "*",8	sucht das erste Programm auf der Diskette.

NEW

Dieses Kommando löscht das Programm im Speicher des Rechners. Alle eventuell benutzten Variablen sind danach ebenfalls gelöscht. Wenn das

Programm nicht durch SAVE gespeichert wurde, ist es verloren.
Gebrauchen Sie diesen Befehl mit Vorsicht.
Sie können NEW auch in einem Programm verwenden, wenn Sie möchten, daß der Computer nach Ausführung dieses Programms wieder frei für neue Programme ist.

RUN

Dieses Kommando startet die Ausführung eines Programms, wenn es in den Speicher des Computers geladen wurde. RUN ohne Angabe einer Zeilennummer beginnt die Ausführung in der Zeile mit der niedrigsten Nummer. Wird nach RUN eine Nummer angegeben, startet das Programm in der bezeichneten Zeile.

RUN	startet das Programm bei der niedrigsten Zeilennummer
RUN 100	startet das Programm in Zeile 100
RUN X	UNDEFINED STATEMENT ERROR. Die Zeilennummer muß existieren und explizit angegeben werden. Variablen sind nicht erlaubt.

SAVE

Dieses Kommando überträgt das Programm aus dem Rechner auf Band oder Diskette. Wenn Sie nur SAVE eintippen und RETURN drücken, wird das Programm auf Band gespeichert. Der Computer kann nicht feststellen, ob sich an dieser Stelle des Bandes bereits ein Programm befindet. Seien Sie also vorsichtig mit Ihren Bändern, Sie könnten sonst ein wertvolles Programm zerstören.
Lassen Sie nach SAVE einen Namen in Anführungszeichen folgen oder eine Stringvariable, dann wird das Programm diesen Namen erhalten und kann so in Zukunft leichter lokalisiert und wiedergefunden werden. Dem Namen kann zusätzlich noch eine Gerätenummer folgen.
Nach der Gerätenummer kann, durch ein Komma getrennt, eine zweite Zahl folgen und zwar eine 0 oder eine 1. Ist diese zweite Nummer eine 1, dann wird der COMMODORE 64 eine Marke für Band-Ende hinter Ihr Programm schreiben. Das signalisiert dem Computer, daß nach einem LOAD nur bis zu dieser Stelle gesucht werden soll. Wenn Sie LOAD befehlen und der Computer entdeckt dieses Zeichen, bevor das gewünschte Programm gefunden wurde, meldet er einen FILE NOT FOUND ERROR.

SAVE	speichert das Programm ohne Namen auf Band.
SAVE "HALLO"	speichert das Programm unter dem Namen HALLO auf Band.
SAVE A$	speichert das Programm auf Band mit dem Namen, der in A$ festgelegt ist.
SAVE "HALLO",8	speichert das Programm unter dem Namen HALLO auf Diskette.
SAVE "HALLO",1,1	speichert das Programm unter dem Namen HALLO auf Kassette und setzt dahinter eine Marke für BAND-ENDE.

VERIFY

Durch VERIFY wird der Computer veranlaßt, das Programm in seinem Speicher mit einem Programm auf Band oder Diskette zu vergleichen. Dies stellt sicher, daß das Programm bei SAVE korrekt weggespeichert wurde und währenddessen kein Fehler auf dem Band oder der Diskette auftrat. VERIFY ohne weitere Angabe läßt den COMMODORE 64 mit dem nächsten Programm auf dem Band vergleichen, unabhängig von dessen Namen.
VERIFY gefolgt von einem Programmnamen oder einer Stringvariablen sucht dieses Programm und prüft es dann. Zusätzlich kann eine Gerätenummer angegeben werden.

VERIFY	prüft das nächste Programm auf Kassette
VERIFY "HALLO"	sucht das Programm HALLO und vergleicht mit dem Programm im Speicher
VERIFY "HALLO",8	sucht auf der Diskette und prüft dann

BEFEHLE

CLOSE

Dieser Befehl vervollständigt Files, die mit OPEN geöffnet wurden und schließt sie wieder. Die Zahl nach CLOSE ist die Nummer des Files, das geschlossen werden soll.

CLOSE 2	Nur File #2 wird geschlossen

CLR

Dieser Befehl löscht alle Variablen im Speicher, läßt aber das Programm selbst intakt. CLR wird automatisch ausgeführt, wenn RUN befohlen wird.

CMD

CMD sendet Ausgaben, die normalerweise auf den Bildschirm gehen würden (z. B. PRINT oder LIST, aber nicht POKE auf den Bildschirm), stattdessen an ein anderes Gerät. Das kann ein Drucker sein oder ein Datenfile auf Band oder Diskette. Das Gerät oder File muß zuvor mit OPEN geöffnet werden. Dem Befehl CMD muß eine Zahl oder numerische Variable folgen, die sich auf die Filenummer bezieht.

OPEN 1,4	öffnet Gerät #4, den Drucker
CMD 1	jede normale Ausgabe soll an den Drucker gehen
LIST	das Programmlisting wird nun nicht auf dem Bildschirm, sondern auf dem Drucker ausgegeben

Um die Ausgabe wieder auf den Bildschirm zu schicken, beenden Sie den CMD 1 Befehl mit PRINT #1 und schließen Sie das File mit CLOSE 1.

DATA

Dieser Aussage folgt eine Liste von Angaben, die durch READ genutzt werden sollen. Die Angaben können von numerischem Typ sein oder Textstrings. Sie werden untereinander durch Kommata getrennt. Strings brauchen nicht in Anführungszeichen zu stehen, es sei denn, sie enthalten Leerstellen, Doppelpunkt oder Komma. Steht einmal nichts zwischen zwei Kommata, dann wird der Wert numerisch als 0 gelesen oder als leerer String.

```
10 DATA 12, 14.5, "HALLO, PARTNER", 3.14 TEIL1
```

DEF FN

Eine komplexe Rechnung kann als Funktion mit einem kurzen Namen definiert werden. Im Falle einer sehr umfangreichen Formel, die häufig benötigt wird, kann damit viel Platz und Zeit eingespart werden. Der Funktionsname ist FN, erweitert um jeden zulässigen Variablennamen (1 oder 2 Zeichen). Die Funktion wird definiert durch das Statement DEF,

dem der Funktionsname folgt. Nach dem Namen wird in Klammern ein numerische Variable angegeben. Danach kommt ein Gleichheitszeichen und dann die aktuelle Formel, mit der Variablen am richtigen Platz. Die Funktion kann nun aufgerufen werden, indem für die Variable jeder beliebige Wert oder eine andere Variable eingesetzt wird.

```
10 DEF FNA(X) = 12*(34.75 – X/.3)
20 PRINT FNA(7)
```

Für dieses Beispiel wird das Ergebnis 137 sein.

DIM (DIMENSIONIEREN EINES FELDES)

Wenn ein Array mehr als 11 Elemente enthalten soll, muß dafür ein DIM Befehl ausgeführt werden. Behalten Sie im Auge, daß das ganze Array Platz im Speicher belegt. Erzeugen Sie also kein Array, das wesentlich größer ist, als Sie unbedingt benötigen.
Um zu ermitteln, wieviele Variablen ein Array enthält, multiplizieren Sie die Größe aller Dimensionen eines Arrays miteinander.

```
10 DIM A$(40), B7(15), CC%(4,4,4)
            ↑          ↑           ↑
       41 Elemente  16 Elemente   125 Elemente
```

Sie können mehr als ein Array in einem DIM Statement erklären. Dimensionieren Sie ein Array aber nicht mehr als einmal.

END

Wenn ein Programm ein END entdeckt, hält es an, als ob es ans Ende der letzten Zeile gekommen wäre. Mit CONT können Sie es wieder starten.

FOR ... TO ... STEP

Zusammen mit dem Befehl NEXT sorgt dieses Statement dafür, daß ein Programmabschnitt mehrmals durchlaufen wird (Die Anzahl der Durchläufe kann angegeben werden). Das Format dieses Befehls ist:

FOR (Variablenname) = (Zähleranfang) TO (Zählerende) STEP (Schrittweite)

Während des Programmlaufs wird zur Laufvariable der als Schrittweite gegebene Wert addiert oder subtrahiert. Wird kein STEP angegeben, bedeutet dies Schrittweite = 1. Zähleranfang und Zählerende stellen die Grenzwerte für die Laufvariable dar.

```
10 FOR L = 1 TO 10 STEP .1
20 PRINT L
30 NEXT L
```

Die Schrittweite kann auch negativ angegeben werden.

GET

Mit GET können Daten von der Tastatur übernommen werden, und zwar jedes Mal ein einzelnes Zeichen. Wenn GET ausgeführt wird, wird das eingetippte Zeichen der Variablen zugeordnet. War kein Zeichen eingetippt, wird ein leeres Zeichen oder Null zugeordnet.
Nach GET steht ein Variablenname, gewöhnlich eine Stringvariable. Wird eine numerische Variable benutzt, und eine nicht numerische Taste gedrückt, hält das Programm mit einer Fehlermeldung an.
Der GET Befehl sollte in einer Schleife formuliert werden, die ein leeres Resultat abprüft. Die unten stehende Schleife wartet, bis irgendeine Taste gedrückt wird:

```
10 GET A$ : IF A$ = "" THEN 10
```

GET#

GET# wird benutzt mit einem vorher durch OPEN geöffneten File oder Gerät, um von diesem ein Zeichen einzulesen.

```
GET#1,A$
```

liest ein Zeichen aus einem Datenfile.

GOSUB

Dieser Befehl wirkt ähnlich wie GOTO, der Computer merkt sich aber, an welcher Stelle des Programms er den Befehl GOSUB erhalten hat. Erreicht das Programm den Befehl RETURN, springt es an die Stelle direkt hinter dem GOSUB zurück. Dies ist besonders nützlich, wenn ein Programmteil häufiger benutzt werden kann. Man muß diese

Programmzeilen nicht mehrfach eintippen, sondern kann sie durch GOSUB bei Bedarf "aufrufen".

```
20 GOSUB 800
```

GOTO oder GO TO

Das Programm springt in die nach GOTO angegebene Zeile und fährt dort mit der Arbeit fort. Kommt die nach GOTO angegebene Zeilennummer nicht im Programm vor, stoppt das Programm mit einer Fehlermeldung.

IF ... THEN

IF ... THEN läßt den Computer die vorliegende Situation analysieren. Er entscheidet sich dann für eine von zwei Möglichkeiten. je nach Resultat der Analyse. Ist die Bedingung nach IF erfüllt (wahr), dann wird der Teil nach THEN ausgeführt. Das kann jedes beliebige Basic Statement sein.
Ist die Bedingung nicht erfüllt, geht das Programm zur nächsten Zeile über. Die Bedingung kann eine Variable oder eine Formel sein. Sie gilt als wahr, wenn das Ergebnis der Berechnung nicht Null ist, anderenfalls gilt sie als nicht erfüllt (falsch). In den meisten Fällen steht als Bedingung ein Ausdruck, der einen der logischen Operatoren (=, <, >, <=, =>, <>, AND, OR, NOT) enthält:

```
10 IF X > 0 THEN END
```

INPUT

Mit dem INPUT Befehl kann sich das Programm Daten vom Benutzer übergeben lassen und sie einer Variablen zuordnen. Das Programm hält an, gibt ein Fragezeichen auf dem Bildschirm aus und wartet, bis der Benutzer eine Antwort eingetippt und die RETURN-Taste gedrückt hat.
Nach INPUT steht ein Variablenname oder eine Liste von Variablen, die durch Kommata getrennt sind. Zwischen INPUT und den Variablen kann in Anführungszeichen eine Botschaft an den Benutzer vorgegeben werden. Werden Eingaben für mehrere Variablen gemacht, müssen diese durch Kommata getrennt eingetippt werden.

```
10 INPUT "BITTE GEBEN SIE IHREN NAMEN AN";A$
20 PRINT "GEBEN SIE JETZT IHRE CODEZAHL EIN" : INPUT B
```

INPUT#

INPUT# ist ähnlich wie INPUT, übernimmt die Daten aber von einem vorher mit OPEN geöffneten File.

LET

LET leitet eine Zuweisung ein, wird aber kaum jemals in Programmen benutzt, da seine Angabe wahlfrei ist. Der Variablenname, dem das Ergebnis einer Rechnung zugewiesen werden soll, steht links vom Gleichheitszeichen, die Formel steht rechts.

```
LET A=5
LET D$="HALLO"
```

NEXT

NEXT wird immer in Verbindung mit FOR... TO benutzt. Erreicht das Programm den Befehl NEXT, überprüft es die Laufvariable, um zu sehen, ob die vorgeschriebene Grenze schon erreicht ist. Ist die Schleife noch nicht zu Ende, wird der nach STEP angegebene Wert zum Zähler hinzuaddiert. Ist die Grenze bereits erreicht, fährt das Programm mit dem auf NEXT folgenden Befehl fort.
Nach NEXT darf ein Variablenname oder eine Liste von Variablen, durch Kommata getrennt, stehen. Es muß immer die innerste Schleife zuerst abgearbeitet werden.

```
10 FOR X=1 TO 100 : NEXT
```

ON

Dieser Befehl macht aus den Kommandos GOTO und GOSUB spezielle Versionen des IF Befehls. Auf ON folgt eine numerische Variable oder eine Formel, die errechnet wird. Ist das Resultat der Rechnung 1, wird die erste Zeile der Liste ausgeführt. Ist das Resultat 2, wird die zweite Zeile ausgeführt usw. Ist das Resultat 0 oder größer als die angeführte Liste von Zeilennummern, wird der Befehl nach ON bearbeitet.

```
10 INPUT X
20 ON X GOTO 10,20,30,40,50
```

OPEN

Mit dem OPEN Befehl kann der COMMODORE 64 Daten mit Geräten wie dem Kassettenrecorder oder Diskettenlaufwerk, einem Drucker oder sogar dem Bildschirm austauschen. Auf das Schlüsselwort OPEN folgt eine Nummer (0-255), die logische Filenummer, auf die sich alle nachfolgenden Befehle beziehen. Gewöhnlich folgt der ersten Nummer eine zweite, die Gerätenummer.

Die Gerätenummern sind:

0	Tastatur
1	Kassette
3	Bildschirm
4	Drucker (oder 5)
8	Disk (oder 9-15)

Der Gerätenummer folgt häufig, ebenfalls durch Komma getrennt, eine dritte Zahl, die Sekundäradresse. Im Falle der Kassette ist dies eine 0 für Lesen, eine 1 für Schreiben und eine 2 für Schreiben mit Markierung für Bandende.
Im Falle der Disk bezieht sich die Sekundäradresse auf die Puffer- oder Kanalnummer. Beim Drucker kontrolliert sie Zusatzeinrichtungen wie etwa formatierten Druck. Weitere Einzelheiten finden Sie im COMMODORE 64 Programmierhandbuch.

OPEN 1,0	öffnet die Tastatur als Gerät
OPEN 2,1,8,"D"	öffnet die Kassette zum Lesen, das gesuchte File ist D.
OPEN 3,4	öffnet den Drucker
OPEN 4,8,15	öffnet den Befehlskanal auf der Disk

Siehe auch: CLOSE, CMD, GET#, INPUT# und PRINT#, Systemvariable ST und Anhang B.

POKE

POKE wird immer von zwei Zahlen oder Formeln gefolgt. Die erste Angabe ist die Adresse eines Speicherplatzes. Die zweite Angabe ist ein Wert zwischen 0 und 255, der in die entsprechende Speicherstelle geschrieben wird und der den bisher dort stehenden Wert ersetzt.

```
10 POKE 53281,0
20 S=4096*13
30 POKE S+29,8
```

PRINT

PRINT ist wohl der erste Befehl, den ein Neuling zu gebrauchen lernt. Einige Variationen sollten dabei beachtet werden. Nach PRINT können stehen:

Textstrings in Anführungszeichen
Variablennamen
Funktionen
Satzzeichen

Satzzeichen dienen der Formatierung der Daten auf dem Bildschirm. Das Komma unterteilt den Bildschirm in vier Spalten, wogegen das Semikolon jeden Zwischenraum unterdrückt. Jedes der beiden Zeichen kann als letztes Symbol in einer Zeile vorkommen. Dies bewirkt, daß das nächste mit PRINT ausgegebene Zeichen so behandelt wird, als sei es eine Fortsetzung des ursprünglichen PRINT Befehls.

```
10 PRINT "HALLO"
20 PRINT "HALLO",A$
30 PRINT A+B
40 PRINT J;
50 PRINT A,B,C,D
```

Siehe auch die Funktionen POS, SPC und TAB

PRINT#

Zwischen diesem Befehl und PRINT bestehen einige Unterschiede. Auf PRINT# folgt eine Zahl, die sich auf ein vorher mit OPEN geöffnetes Gerät oder Datenfile bezieht. Nach dieser Zahl steht ein Komma und dann die Liste dessen, was auszugeben ist. Das Komma und das Semikolon haben den selben Effekt wie bei PRINT. Beachten Sie, daß einige Geräte nichts mit TAB oder SPC anfangen können.

Vorsicht: PRINT# darf nicht durch ?# abgekürzt werden!

```
100 PRINT#1,"DATEN INHALTE";A%,B1,C$
```

READ

READ wird benutzt, um Informationen aus DATA Zeilen auf Variablen zu übertragen, damit diese verarbeitet werden können. Sie müssen darauf achten, daß kein String gelesen wird, wenn READ einen numerischen Wert erwartet. Dies würde einen TYPE MISMATCH ERROR ergeben.

REM (Remark, Bemerkung)

Nach REM steht eine Bemerkung für irgend jemanden, der das Programmlisting lesen wird. Sie könnte zum Beispiel einen Programmabschnitt erklären oder zusätzliche Informationen beinhalten. Hinter REM kann ein beliebiger Text stehen. Es hat auf ein Programm keinerlei Wirkung, außer daß es dieses länger macht.

RESTORE

RESTORE setzt den Zeiger auf das nächste aus einer DATA Zeile zu lesende Element auf den Anfang zurück. Diese Informationen können so mehrfach gelesen werden.

RETURN

Dieser Befehl schließt ein Unterprogramm ab. Wenn das Programm ein RETURN entdeckt, springt es zu dem unmittelbar auf GOSUB folgenden Befehl zurück. Wurde zuvor kein GOSUB befohlen, erhalten Sie einen RETURN WITHOUT GOSUB ERROR.

STOP

Dieser Befehl hält ein Programm an. Die Meldung BREAK IN xxx wird angezeigt, wobei xxx die Zeilennummer ist, in der STOP steht. Das Programm kann nach Eingabe von CONT weiter arbeiten. STOP wird gewöhnlich beim Debugging, der Suche nach Programmierfehlern, gebraucht.

SYS

Auf SYS folgt eine Zahl oder ein numerischer Ausdruck im Bereich 0-65535. Das Programm startet dann ein Programm in Maschinensprache, das bei dieser Speicheradresse beginnt. SYS ist ähnlich dem USR, läßt aber keine Übergabe von Parametern zu.

WAIT

WAIT wird benutzt, um ein Programm solange anzuhalten, bis eine bestimmte Speicheradresse einen bestimmten Wert angenommen hat.

Auf WAIT folgt eine Speicheradresse (X) und bis zu zwei Variablen. Das Format ist:

WAIT X,Y,Z

Der Inhalt der angegebenen Adresse wird zuerst EXCLUSIV OR verknüpft mit dem dritten Wert Z (falls dieser angegeben ist), und dann wird mit dem zweiten Wert ein logisches AND ausgeführt. Ist das Resultat 0, prüft das Programm den Inhalt dieser Adresse erneut. Ist das Ergebnis nicht Null, arbeitet das Programm mit dem nächsten Statement weiter.

NUMERISCHE FUNKTIONEN

ABS(X) (absoluter Wert)

ABS ergibt den absoluten Wert einer Zahl ohne deren Vorzeichen (+ oder –). Die Antwort ist immer positiv oder 0.

ATN(X) (Arcustangens)

Ergibt den Winkel (im Bogenmaß), dessen Tangens X ist.

COS(X) (Cosinus)

ergibt den Cosinus des Winkels X. Der Winkel ist im Bogenmaß anzugeben.

EXP(X)

Ergibt die X-te Potenz der mathematischen Konstanten e (2.71827183).

FNxx(X)

Ergibt das Resultat einer benutzer-definierten Funktion FNxx, die in einem DEFFN Statement festgelegt wurde.

INT(X)

Ergibt den Vorkommateil von X. Alle Stellen nach dem Dezimalpunkt werden abgeschnitten. Das Ergebnis ist immer kleiner oder gleich X. Das

bedeutet, daß negative Zahlen dem Betrag nach größer werden.
(INT(– 2.1) = – 3)

LOG(X)

Ergibt den natürlichen Logarithmus von X zur Basis e (siehe EXP(X)). Zur Umwandlung in den Zehnerlogarithmus wird durch LOG(10) dividiert.

PEEK(X)

Ergibt den Inhalt der Speicheradresse X, mit X im Bereich 0-65535. Das Ergebnis ist eine ganze Zahl im Bereich 0-255. PEEK wird oft im Zusammenhang mit POKE gebraucht.

RND(X) (random number, Zufallszahl)

RND(X) ergibt eine Zufallszahl zwischen 0 und 1. Die erste Zufallszahl sollte durch die Formel RND(– TI) erzeugt werden, damit sie bei jedem Programmlauf eine neue Startzahl bekommen. Später sollte X = 0 oder eine positive Zahl sein.
Ein negativer Parameter X bildet eine neue Startzahl für den Zufallszahlen-Generator. Dieselbe negative Zahl wird immer dieselbe Folge von "Zufallszahlen" ergeben, falls diese mit RND(1) gebildet wird. RND(0) liefert stets neue Folgen.
Eine Zufallszahl zwischen X und Y erhält man nach der Formel:

N = INT(RND(1)*Y)+X

wobei X die untere Grenze Y die Obergrenze der gewünschten Zahlenbereiche ist.

SGN(X) (Signum, Vorzeichen)

Mit dieser Funktion ermittelt man das Vorzeichen von X. Das Ergebnis ist 1, wenn X positiv ist, 0, wenn X=0 und – 1, wenn X negativ ist.

SIN(X) (Sinus)

Ergibt den Sinus des Winkels X. Der Winkel ist im Bogenmaß anzugeben.

SQR(X) (Quadratwurzel)

Ergibt die Quadratwurzel von X, für X größer oder gleich 0. Bei negativem X erhält man die Fehlermeldung ILLEGAL QUANTITY ERROR.

TAN(X) (Tangens)

Das Ergebnis ist der Tangens des Winkels X, mit X im Bogenmaß.

USR(X)

Wenn die Funktion USR benutzt wird, springt das Programm in ein Maschinenprogramm, dessen Startadresse in den Speicherstellen 785 und 786 abgelegt ist. Der Parameter X wird an das Maschinenprogramm übergeben, das selbst einen anderen Wert an das BASIC-Programm zurückgibt. Weitere Einzelheiten zu dieser Funktion und zur Programmierung in Maschinensprache finden Sie im COMMODORE 64 Programmierhandbuch.

STRING FUNKTIONEN

ASC(X$)

Ergibt den ASCII-Code des ersten Zeichens von X$.

CHR$(X)

Dies ist die Umkehrfunktion zu ASC(X$) und ergibt das Zeichen, dessen ASCII-Code X ist.

LEFT$(X$,X)

Ergibt einen Teilstring, der die linken X Zeichen von X$ enthält.

LEN(X$)

Ergibt die Anzahl Zeichen (einschließlich Leerzeichen und anderer Symbole) in X$.

MID$(X$,S,X)

Ergibt einen String, der X Zeichen von X$ enthält, beginnend mit dem S-ten Zeichen.

RIGHT$(X$, X)

Ergibt die rechten X Zeichen von X$.

STR$(X)

Ergibt einen String, der mit der durch PRINT ausgegebenen Version von X identisch ist.

VAL(X$)

Diese Funktion wandelt X$ in eine Zahl um und ist im Wesentlichen die Umkehrfunktion von STR$(X). Der String wird vom ersten linken Zeichen an nach rechts untersucht bis das erste Zeichen gefunden wird, das nicht mehr in ein Zahlenformat paßt. Die so ermittelte Zeichenfolge wird in eine Zahl umgewandelt.

X = VAL ("123.456")	X = 123.456
X = VAL ("12A13B")	X=12
X = VAL ("RIU017")	X=0
X = VAL ("– 1.23.15.67")	X = – 1.23

ANDERE FUNKTIONEN

FRE(X)

Diese Funktion ergibt die Anzahl freier Bytes im Speicher, unabhängig vom Wert von X.

POS(X)

Diese Funktion ergibt die Position in der Bildschirm-Zeile (0-39) an der das nächste PRINT Statement ausgeführt würde. X kann jeden Wert annehmen und wird nicht benutzt.

SPC(X)

Diese Funktion wird in PRINT Befehlen benutzt, um X Zeichen zu überspringen.

TAB(X)

TAB wird ebenfalls mit PRINT benutzt. Das nächste Zeichen wird in Spalte X gedruckt. (Wird vom Drucker wie SPC(X) behandelt.)

ABKÜRZUNGEN DER BASIC SCHLÜSSELWÖRTER

Damit Sie beim Eintippen Ihrer Programme und Direktkommandos Zeit sparen können, dürfen Sie die meisten Schlüsselwörter im BASIC des COMMODORE 64 abkürzen. Die Abkürzung für PRINT ist das Fragezeichen. Die Abkürzungen für andere Schlüsselwörter werden gebildet, indem man die ersten ein oder zwei Zeichen eintippt und das nächste Zeichen des Wortes mit SHIFT eingibt. Werden die Abkürzungen in Programmzeilen benutzt, gibt LIST die Schlüsselwörter in der ausgeschriebenen Form aus. Beachten Sie, daß Abkürzungen für einige Schlüsselwörter bereits eine linke Klammer mit einschließen.

Befehl	Abkürzung	wird angezeigt als	Befehl	Abkürzung	wird angezeigt als
ABS	A **<SHIFT>** B	A	FOR	F **<SHIFT>** O	F
AND	A **<SHIFT>** N	A	FRE	F **<SHIFT>** R	F
ASC	A **<SHIFT>** S	A	GET	G **<SHIFT>** E	G
ATN	A **<SHIFT>** T	A	GOSUB	GO **<SHIFT>** S	GO
CHR$	C **<SHIFT>** H	C	GOTO	G **<SHIFT>** O	G
CLOSE	CL **<SHIFT>** O	CL	INPUT#	I **<SHIFT>** N	I
CLR	C **<SHIFT>** L	C	LET	L **<SHIFT>** E	L
CMD	C **<SHIFT>** M	C	LEFT$	LE **<SHIFT>** F	LE
CONT	C **<SHIFT>** O	C	LIST	L **<SHIFT>** I	L
DATA	D **<SHIFT>** A	D	LOAD	L **<SHIFT>** O	L
DEF	D **<SHIFT>** E	D	MID$	M **<SHIFT>** I	M
DIM	D **<SHIFT>** I	D	NEXT	N **<SHIFT>** E	N
END	E **<SHIFT>** N	E	NOT	N **<SHIFT>** O	N
EXP	E **<SHIFT>** X	E	OPEN	O **<SHIFT>** P	O

Befehl	Abkürzung	wird angezeigt als	Befehl	Abkürzung	wird angezeigt als
PEEK	P **<SHIFT>** E	P	SPC(	S **<SHIFT>** P	S
POKE	P **<SHIFT>** O	P	SQR	S **<SHIFT>** Q	S
PRINT	?	?	STEP	ST **<SHIFT>** E	ST
PRINT#	P **<SHIFT>** R	P	STOP	S **<SHIFT>** T	S
READ	R **<SHIFT>** E	R	STR$	ST **<SHIFT>** R	ST
RESTORE	RE **<SHIFT>** S	RE	SYS	S **<SHIFT>** Y	S
RETURN	RE **<SHIFT>** T	RE	TAB	T **<SHIFT>** A	T
RIGHT$	R **<SHIFT>** I	R	THEN	T **<SHIFT>** H	T
RND	R **<SHIFT>** N	R	USR	U **<SHIFT>** S	U
RUN	R **<SHIFT>** U	R	VAL	V **<SHIFT>** A	V
SAVE	S **<SHIFT>** A	S	VERIFY	V **<SHIFT>** E	V
SGN	S **<SHIFT>** G	S	WAIT	W **<SHIFT>** A	W
SIN	S **<SHIFT>** I	S			

BILDSCHIRM CODES

Die folgende Tabelle listet alle in den Zeichensätzen des COMMODORE 64 vorkommenden Zeichen auf. Sie zeigt, welche Zahl in den Bildschirmspeicher (Adressen 1024-2023) gePOKEd werden muß, um ein bestimmtes Zeichen zu erhalten. Sie zeigt umgekehrt, welches Zeichen einer aus dem Bildschirmspeicher gePEEKten Zahl entspricht.
Zwei Zeichensätze sind verfügbar, doch immer nur einer zu einer Zeit. Das heißt, Sie können nicht Zeichen eines Satzes auf den Bildschirm bringen, während auch Zeichen des anderen Satzes angezeigt werden. Zwischen den Zeichensätzen wird umgeschaltet, indem man die SHIFT-Taste und die COMMODORE-Taste gleichzeitig drückt.
Von Basic können Sie mit POKE 53272,21 in den Groß-/Grafik-Modus und mit POKE 53272,23 in den Groß-/Klein-Modus umschalten.
Jedes Zeichen der Tabelle kann auch in REVERS (dunkel auf hellem Grund) dargestellt werden, indem Sie 128 zum angegebenen Wert addieren.
Wollen Sie zum Beispiel in 1504 einen ausgefüllten Kreis darstellen, POKEn Sie in diese Adresse den Code (81) des ausgefüllten Kreises: POKE 1504,81.
Zu jedem Platz des Bildschirmspeichers gibt es eine korrespondierende Adresse (55296-56295), die die Farbe des Zeichens kontrolliert. Um die Farbe des Kreises in gelb (Farbcode 7) zu ändern, POKEn Sie die entsprechende Adresse (55776) mit dem Farbwert: POKE 55776,7.
In Anhang G finden Sie die vollständigen Adressätze des Bildschirm- und Farbspeichers, zusammen mit den Farbcodes.

BILDSCHIRM CODES

Satz 1	Satz 2	Poke	Satz 1	Satz 2	Poke	Satz 1	Satz 2	Poke
@		0	V	v	22	,		44
A	a	1	W	w	23	-		45
B	b	2	X	x	24	.		46
C	c	3	Y	y	25	/		47
D	d	4	Z	z	26	0		48
E	e	5	[		27	1		49
F	f	6	£		28	2		50
G	g	7	]		29	3		51
H	h	8	↑		30	4		52
I	i	9	←		31	5		53
J	j	10	**<SPACE>**		32	6		54
K	k	11	!		33	7		55
L	l	12	"		34	8		56
M	m	13	#		35	9		57
N	n	14	$		36	:		58
O	o	15	%		37	;		59
P	p	16	&		38	<		60
Q	q	17	'		39	=		61
R	r	18	(		40	>		62
S	s	19	)		41	?		63
T	t	20	*		42			
U	u	21	+		43			

Satz 1	Satz 2	Poke	Satz 1	Satz 2	Poke	Satz 1	Satz 2	Poke
		64		V	86			108
	A	65		W	87			109
	B	66		X	88			110
	C	67		Y	89			111
	D	68		Z	90			112
	E	69			91			113
	F	70			92			114
	G	71			93			115
	H	72			94			116
	I	73			95			117
	J	74	**<SPACE>**		96			118
	K	75			97			119
	L	76			98			120
	M	77			99			121
	N	78			100			122
	O	79			101			123
	P	80			102			124
	Q	81			103			125
	R	82			104			126
	S	83			105			127
	T	84			106			
	U	85			107			

Die Codes 128 - 255 ergeben die invers dargestellten Zeichen der Codes 0 - 127.

An den freien Stellen in Spalte 2 stimmen die beiden Sätze überein.

ANHANG F

ASCII UND CHR$ CODES

Dieser Anhang zeigt für alle x, welches Zeichen auf dem Bildschirm erscheint, wenn Sie PRINT CHR$(x) senden. Umgekehrt können Sie auch das Ergebnis für PRINT ASC("x") entnehmen, wobei "x" der zugehörigen gedrückten Taste entspricht. Dies ist besonders nützlich, um das in einem GET Statement empfangene Zeichen zu ermitteln, zwischen Groß- und Kleinschrift umzuschalten und um nicht druckbare Steuerzeichen, wie etwa Umschaltung Groß-/Kleinschrift zu senden, die nicht in einem String zwischen Anführungszeichen eingeschlossen werden können.

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	0	**<CRSR unten>**	17	"	34	3	51
	1	**<REVERS ein>**	18	#	35	4	52
	2	<HOME>	19	$	36	5	53
	3	<DEL>	20	%	37	6	54
	4		21	&	38	7	55
<weiss>	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
blockiert **<SHIFT> <C=>**	8		25	*	42	;	59
entriegelt **<SHIFT> <C=>**	9		26	+	43	<	60
	10		27	,	44	=	61
	11	<rot>	28	-	45	>	62
	12	<CRSR rechts>	29	.	46	?	63
<RETURN>	13	<grün>	30	/	47	@	64
UMSCHALTUNG KLEINBUCHST.	14	<blau>	31	0	48	A	65
	15	<SPACE>	32	1	49	B	66
	16	!	33	2	50	C	67

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
D	68		97		126	<grau 3>	155
E	69		98		127	<violett>	156
F	70		99		128	<CRSR links>	157
G	71		100	<orange>	129	<gelb>	158
H	72		101		130	<türkis>	159
I	73		102		131	<SPACE>	160
J	74		103		132		161
K	75		104	<f 1>	133		162
L	76		105	<f 3>	134		163
M	77		106	<f 5>	135		164
N	78		107	<f 7>	136		165
O	79		108	<f 2>	137		166
P	80		109	<f 4>	138		167
Q	81		110	<f 6>	139		168
R	82		111	<f 8>	140		169
S	83		112	<SHIFT> <RETURN>	141		170
T	84		113	UMSCHALTUNG GROSSBUCHST.	142		171
U	85		114		143		172
V	86		115	<schwarz>	144		173
W	87		116	<CRSR oben>	145		174
X	88		117	<revers aus>	146		175
Y	89		118	<CLR>	147		176
Z	90		119	<INST>	148		177
[	91		120	<braun>	149		178
£	92		121	<hellrot>	150		179
]	93		122	<grau 1>	151		180
↑	94		123	<grau 2>	152		181
←	95		124	<hellgrün>	153		182
	96		125	<hellblau>	154		183

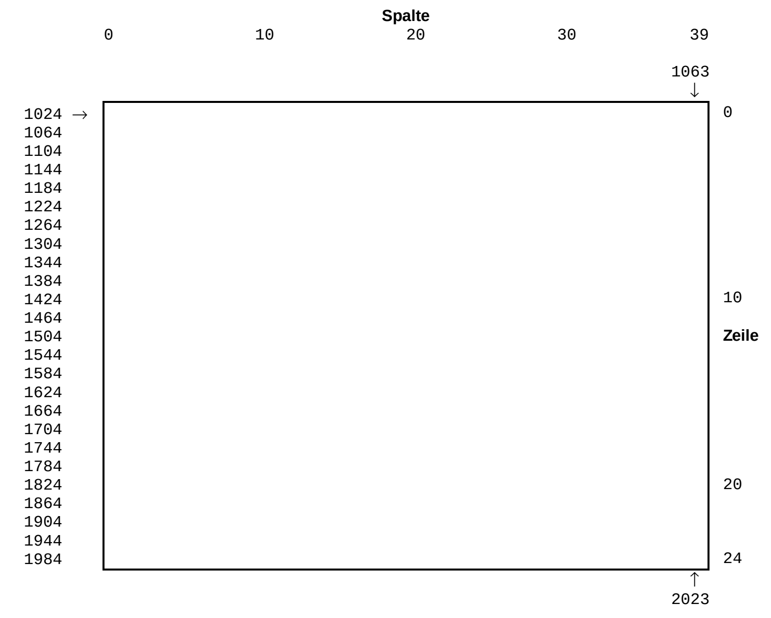
Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	184		186		188		190
	185		187		189		191

Codes	192 - 223	wie Codes	96 - 127
Codes	224 - 254	wie Codes	160 - 190
Code	255	wie Code	126

ADRESSBEREICHE VON BILDSCHIRM- UND FARBSPEICHER

Die folgenden Tabellen und Diagramme zeigen, welche Adressen die auf dem Bildschirm dargestellten Zeichen und deren individuelle Farben kontrollieren, sowie die Farbcodes.

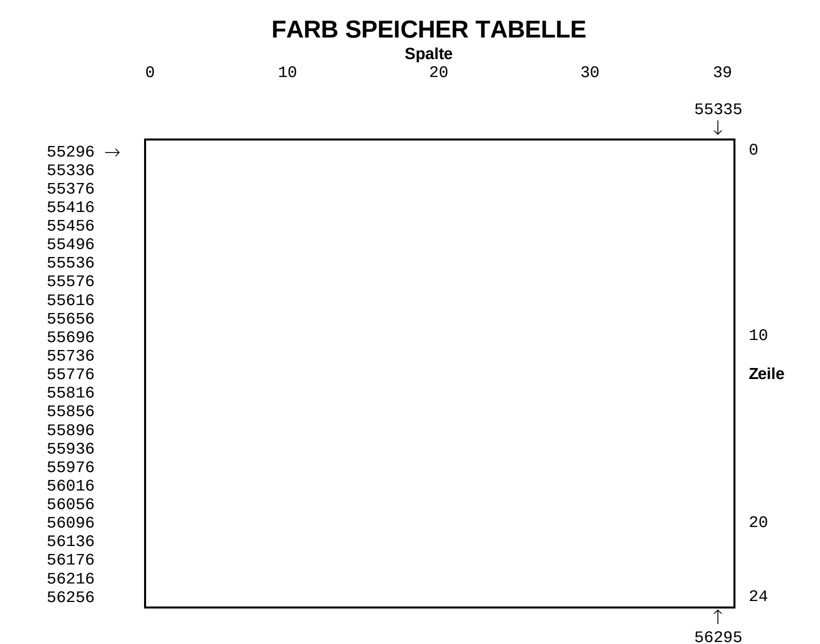
BILDSCHIRM SPEICHER TABELLE



Um ein Zeichen in einer bestimmten Farbe darzustellen, sind die folgenden Werte in die entsprechende Adresse des Farbspeichers zu POKEn:

0	SCHWARZ	8	ORANGE
1	WEISS	9	BRAUN
2	ROT	10	HELLROT
3	TÜRKIS	11	GRAU 1
4	VIOLETT	12	GRAU 2
5	GRÜN	13	HELLGRÜN
6	BLAU	14	HELLBLAU
7	GELB	15	GRAU 3

Um zum Beispiel die Farbe des Zeichens in der linken oberen Ecke des Bildschirms nach rot zu ändern, geben Sie ein: POKE 55296,2



ABGELEITETE MATHEMATISCHE FUNKTIONEN

Funktionen, die in Commodore 64 Basic nicht vordefiniert sind, können mit Hilfe der folgenden Formeln berechnet werden:

FUNKTION	BERECHNUNG BASIC
SEKANS	SEC(X) = 1/COS(X)
COSEKANS	CSC(X) = 1/SIN(X)
COTANGENS	COT(X) = 1/TAN(X)
ARCUSSINUS	ARSIN(X) = ATN(X/SQR(1–X↑2))
ARCUCOSINUS	ARCOS(X) = –ATN(X/SQR(1–X↑2))+π/2
ARCUSCOTANGENS	ARCOT(X) = ATN(X)+π/2
ARCUSSEKANS	ARSEC(X) = ATN(X/SQR(X↑2–1))
ARCUSCOSECANS	ARCSC(X) = ATN(X/SQR(X↑2–1))+(SGN(X)–1)*π/2
SINUS HYPERBOLICUS	SINH(X) = (EXP(X)–EXP(–X))/2
COSINUS HYPERBOLICUS	COSH(X) = (EXP(X)+EXP(–X))/2
TANGENS HYPERBOLICUS	TANH(X) = EXP(–X)/(EXP(X)+EXP(–X))*2+1
CONTANGENS HYPERBOLICUS	COTH(X) = EXP(–X)/(EXP(X)–EXP(–X))*2+1
SEKANS HYPERBOLICUS	SECH(X) = 2/(EXP(X)+EXP(–X))
COSEKANS HYPERBOLICUS	CSCH(X) = 2/(EXP(X)–EXP(–X))
ARCUSSINUS HYPERBOLICUS	ARSINH(X) = LOG(X+SQR(X↑2+1))
ARCUSCOSINUS HYPERBOLICUS	ARCOSH(X) = LOG(X+SQR(X↑2–1))
ARCUSTANGENS HYPERBOLICUS	ARTANH(X) = LOG((1+X)/(1–X))/2
ARCUSCOTANGENS HYPERBOLICUS	ARCOTH(X) = LOG((X+1)/(X–1))/2
ARCUSSEKANS HYPERBOLICUS	ARSECH(X) = LOG((SQR(1–X↑2)+1)/X)
ARCUSCOSEKANS HYPERBOLICUS	ARCSCH(X) = LOG((SQR(1+X↑2)+1)/X)*SGN(X)

ANHANG I

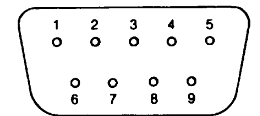
STECKERBELEGUNG DER EIN-/AUSGABE ANSCHLÜSSE

Dieser Anhang soll Ihnen zeigen, wie welches Gerät wo an den COMMODORE 64 angeschlossen werden kann.

1) Steuereingänge für Spiele
2) Modul-Steckplatz
3) Audio/Video
4) Serielle E/A (Disk/Drucker)
5) Kassette
6) User Port

Control Port 1

Pin	Signal	Bem.
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY**	
6	BUTTON A/LP*	
7	+5V	MAX. 100mA
8	GND	
9	POT AX**	



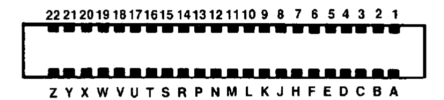
Control Port 2

Pin	Signal	Bem.
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY**	
6	BUTTON B	
7	+5V	MAX. 100mA
8	GND	
9	POT BX**	

*) Button = Feuerknopf am Joystick
LP = Light pen
**) Paddle Potentiometer

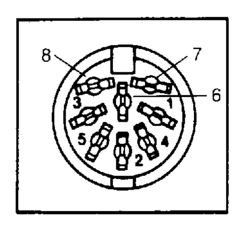
Modul-Steckplatz

Pin	Signal	Pin	Signal
22	GND	Z	GND
21	CD0	Y	CA0
20	CD1	X	CA1
19	CD2	W	CA2
18	CD3	V	CA3
17	CD4	U	CA4
16	CD5	T	CA5
15	CD6	S	CA6
14	CD7	R	CA7
13	-DMA-	P	CA8
12	BA	N	CA9
11	-ROML-	M	CA10
10	-I/O 2-	L	CA11
9	-EXROM-	K	CA12
8	-GAME-	J	CA13
7	-I/O 1-	H	CA14
6	Dot Clock	F	CA15
5	CR/-W-	E	∅2
4	-IRQ-	D	-NMI-
3	+5V	C	-RESET-
2	+5V	B	-ROMH-
1	GND	A	GND



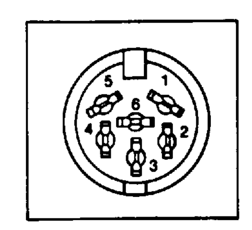
Audio/Video

Pin	Signal	Pin	Signal
1	LUMINANCE	6	CROMINANZ
2	GND	7	O. BELEGUNG
3	AUDIO OUT	8	O. BELEGUNG
4	VIDEO OUT		
5	AUDIO IN		



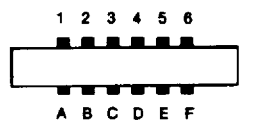
Serielle E/A

Pin	Signal
1	SERIAL -SRQIN-
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	-RESET-



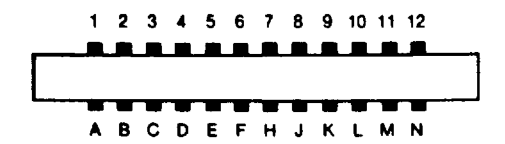
Cassette

Pin	Signal
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



User Port

Pin	Signal	Bem.
1	GND	
2	+5V	MAX. 100mA
3	-RESET-	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	-PC2-	
9	SER. ATN IN	
10	9 VAC	MAX. 100mA
11	9 VAC	MAX. 100mA
12	GND	
A	GND	
B	-FLAG2-	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



PROGRAMME, DIE SIE AUSPROBIEREN SOLLTEN

Wir haben hier für Sie einige Programme aufgelistet, die Sie auf Ihrem Commodore 64 zur Übung eingeben können. Sie zeigen Ihnen Unterhaltendes aber auch Nützliches.

```
100 PRINT"{CLEAR}{DOWN}{REVERSE ON}{RIGHT*6}{SPACE*7}KODEKNACKER{SPACE*10}"
110 INPUT"{DOWN}SPIELANLEITUNG (J/N)";Z$:IFASC(Z$)=78THEN170
120 PRINT"{DOWN}DU MUSST DAS GEHEIME KODE-WORT FINDEN!"
130 PRINT"{DOWN}ES BESTEHT AUS 5 BUCHSTABEN."
140 PRINT"{DOWN}NACH JEDEM VERSUCH ERFAEHRST DU,"
150 PRINT"{DOWN}WIEVIELE BUCHSTABEN SCHON STIMMEN."
160 PRINT"{DOWN}EIN TIP: NICHT EINFACH WILD RUMRATEN!"
170 DATA CJFOF,LBUCF,UBOOF,XJFTF,CMVNF
180 DATA WBUFS,LBUFS,MFEFS,QGFSE,TUBMN
190 DATA TUSPI,MFISF,TUPGG,OVQQF,MFJOF
200 DATA MJTUF,LBOOF,SJOOF,TVQQF,MJFCF
210 DATA SFJTF,NFJTF,HBSCF,GBSCF,MJFCF
220 DATA CVOHF,MVOHF,KVOHF,CBOHF,MBOHF
230 DATA IFUCF,LFUUF,SBUUF,NFJTF,UBTUF
240 DATA EBOLF,QBSLB,UJTDI,TUVIM,IBBSF
250 DATA TQJFM,TQVSU,TQPSU,LBNQG,XFUUF
260 DATA CVDIF,CJSLF,QFTQF,IBGFS,GFSOF
270 N=50
280 DIMN$(N),Z(5),Y(5)
290 FORJ=1TON:READN$(J):NEXTJ
300 T=TI
310 T=T/1000:IFT>=1THEN310
320 Z=RND(-T)
330 G=0:N$=N$(RND(1)*N+1)
340 PRINT"{DOWN}{REVERSE ON}KNACK MEINEN GEHEIM-KODE!":IFR>0THEN380
350 PRINT"{DOWN}ES IST EIN SINNVOLLES 5-BUCHSTABENWORT."
360 PRINT"{DOWN}DU DARFST RATEN, UND ICH SAGE DIR,"
370 PRINT"{DOWN}WIEVIELE BUCHSTABEN STIMMEN."
380 G=G+1:INPUT" DEIN VERSUCH ";Z$
390 IF LEN(Z$)<>5 THENPRINT"{REVERSE ON}5 BUCHSTABEN!":GOTO380
400 V=0:M=0:H=0
410 FORJ=1TO5
420 Z=ASC(MID$(Z$,J,1)):Y=ASC(MID$(N$,J,1))-1:IFY=64THENY=90
430 IFZ<65ORZ>90THENPRINT" {REVERSE ON}UNGUELTIGER KODE!":GOTO380
440 IFZ=65ORZ=69ORZ=73ORZ=79ORZ=85ORZ=89THENV=V+1
450 IFZ=YTHENM=M+1
460 Z(J)=Z:Y(J)=Y:NEXTJ
470 IFM=5THEN580
480 IFV=0ORV=5THENPRINT"{REVERSE ON}WAS SOLL DER QUATSCH?":GOTO380
490 FORJ=1TO5:Y=Y(J)
500 FORK=1TO5:IFY=Z(K)THENH=H+1:Z(K)=0:GOTO520
510 NEXTK
520 NEXTJ
530 PRINT"{UP}{RIGHT*22}";H;"TREFFER"
540 IFG<30THEN380
550 PRINT" {REVERSE ON}GIB'S AUF ! DER KODE HEISST '";
560 FORJ=1TO5:PRINTCHR$(Y(J));:NEXTJ
570 PRINT"'":GOTO590
580 PRINT" DU HAST ES IN"G"VERSUCHEN GESCHAFFT!"
590 INPUT"{REVERSE ON}EIN NEUER KODE";Z$
600 R=1:IFASC(Z$)<>78THEN330
```

```
100 REM MICHAEL ROW THE BOAT ASHORE
110 SI=54272:W1=SI+4:W2=SI+11:W3=SI+18
120 F0(0)=3000:F0(1)=1508:F0(2)=6030
130 D0=180
140 HT=2^(1/12) :REM HALBTONSCHRITT
150 DEFFNH(F)=INT(F/256):REM HI-BYTE
160 DEFFNL(F)=F-256*FNH(F):REM LO-BYTE
170 FORK=0TO20:READX:POKESI+K,X:NEXT:REM KLANGEINSTELLUNGEN
180 READWA,WB,WC:REM WELLENFORMEN
190 DIMF(100,6)
200 PRINT"{CLEAR} STIMME 1  STIMME 2  STIMME 3  DAUER"
210 N=N+1:FOR K=0TO2:READF :REM NOTE AUSDATA
220 F=INT(F0(K)*HT^F+.5):HB=FNH(F):LB=FNL(F):PRINTHB;LB,:REM HI UND LO BYTE
230 F(N,2*K)=LB:F(N,2*K+1)=HB
240 NEXTK
250 READD:PRINT"  "D:F(N,6)=D:REM DAUER
260 IFD>0THEN210
270 POKESI+24,15
280 FORI=1TON-1
290 FORK=0TO2:POKESI+7*K,F(I,2*K):POKESI+7*K+1,F(I,2*K+1):NEXTK:REM 3 FREQ.
300 POKEW1,WA:POKEW2,WB:POKEW3,WC
310 FORK=1TOF(I,6)*D0-150:NEXT
320 POKEW1,0:POKEW2,0:POKEW3,0:REM GENERATOREN AUSSCHALTEN
330 FORK=1TO50:NEXT
340 NEXTI
350 GOTO 280
360 DATA0,0,0,1,0,0,255 :REM SID REGISTER
370 DATA0,0,0,1,0,0,255
380 DATA0,0,0,1,0,0,255
390 DATA37,33,33:REM WELLENFORM
400 DATA0,0,0,2 :REM MELODIE
410 DATA4,0,0,2
420 DATA7,4,0,3
430 DATA 4,0,0,1
440 DATA 7,4,0,2
450 DATA 9,5,0,2
460 DATA 7,4,-5,3
470 DATA -99,-99,-99,1
480 DATA 4,0,0,2
490 DATA 7,0,0,2
500 DATA 9,5,5,8
510 DATA 7,4,0,3
520 DATA -99,-99,-99,1
530 DATA 4,0,0,2
540 DATA 7,4,0,2
550 DATA 7,4,0,3
560 DATA 4,0,0,1
570 DATA 5,2,0,2
580 DATA 4,0,0,2
590 DATA 2,-1,-5,3
600 DATA -99,-99,-99,1
610 DATA 0,-5,0,2
620 DATA 2,-1,0,2
630 DATA 4,0,0,4
640 DATA 2,-5,-5,4
650 DATA 0,0,0,3
660 DATA-99,-99,-99,1
670 DATA -99,-99,-99,-1
```

```
100 REM ALFABET
110 DIMA$(26)
120 Z$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
130 Z1$="12345678901234567890123456"
140 PRINT"{CLEAR}ZIEL DES SPIEL IST ES, EINE FOLGE VON"
150 PRINT"{DOWN}BUCHSTABEN ALFABETISCH ZU ORDNEN."
160 PRINT"{DOWN}DAZU DUERFEN SIE JEWEILS DEN ANFANG"
170 PRINT"{DOWN}DER FOLGE UMDREHEN. VIEL ERFOLG!"
180 PRINT"{DOWN*2}WIE LANG SOLL DIE FOLGE SEIN {DOWN}"
190 INPUT"MAXIMAL 26 ";S
200 IFS<1ORS>26THEN180
210 FORI=1TOS:A$(I)=MID$(Z$,I,1):NEXTI
220 REM ZUFALLSVERTAUSCHUNG
230 FORI=1TOS:K=INT(RND(1)*S+1)
240 T$=A$(I):A$(I)=A$(K):A$(K)=T$
250 NEXTI
260 GOSUB470:T=0
270 REM TEILSTRING UMDREHEN
280 T=T+1
290 INPUT"WIE VIELE SOLL ICH UMDREHEN";R%
300 IFR%=0THEN430
310 IFR%>0ANDR%<=STHEN330
320 PRINT"MUSS ZWISCHEN 1 UND"S"LIEGEN!":GOTO290
330 R=INT(R%/2):FORI=1TOR
340 T$=A$(I)
350 T$=A$(I):A$(I)=A$(R%-I+1):A$(R%-I+1)=T$
360 NEXTI:GOSUB470
370 REM REIHENFOLGE PRUEFEN
380 C=1:FORI=2TOS
390 IFA$(I)>A$(I-1)THEN410
400 C=0
410 NEXTI
420 IFC=0THEN270
430 PRINT"{DOWN}{REVERSE ON}IN"T"{LEFT} ZUEGEN GESCHAFFT!"
440 INPUT"{DOWN}NOCHMAL";Y$
450 IFLEFT$(Y$,1)="J"ORY$="OK"ORY$="1"THEN180
460 END
470 PRINT:PRINTLEFT$(Z1$,S)
480 FORI=1TOS:PRINTA$(I);:NEXTI
490 PRINT"{DOWN}":RETURN
```

```
100 REM ELEKTRONISCHES KLAVIER
110 PRINT"{CLEAR} {REVERSE ON} {RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT}
{221} {RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT} {221} "
120 PRINT" {REVERSE ON} {RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT} {221}
{RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT} {221} "
130 PRINT" {REVERSE ON} {RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT} {221}
{RIGHT} {RIGHT} {221} {RIGHT} {RIGHT} {RIGHT} {221} "
140 PRINT" {REVERSE ON} {221} {221} {221} {221} {221} {221} {221} {221} {221}
{221} {221} {221} {221} {221} "
150 PRINT" {REVERSE
ON}Q{221}W{221}E{221}R{221}T{221}Y{221}U{221}I{221}O{221}P{221}@{221}*{221}^{
221};{221}="
160 PRINT"{DOWN}    'LEERTASTE':{SPACE*5}PEDAL"
170 PRINT"{DOWN}'F1','F3','F5','F7': OKTAVE"
180 PRINT"{DOWN}'F2','F4','F6','F8': WELLENFORM"
190 PRINT"   {DOWN}GEDULD, ICH MUSS RECHNEN!"
200 SI=13*4096+1024:DIMF(26):DIMK(255)
210 FORI=0TO28:POKESI+I,0:NEXT
220 F1=7939:FORI=1TO26:F(27-I)=F1*5.8+30:F1=F1/2^(1/12):NEXT
230 K$="1Q2W3ER5T6Y7UI9O0P@-*\^{HOME};=":REM BENUETZTE TASTEN
240 FORI=1TOLEN(K$):K(ASC(MID$(K$,I)))=I:NEXT
250 PRINT"{UP}{SPACE*31}"
260 AN=0:AB=0:HA=15:AU=9:HH=HA*16+AU:AS=AN*16+AB:WF=16:M=1:OK=4:HB=256:Z=0
270 FORI=0TO2:POKESI+5+I*7,AS:POKESI+6+I*7,HT
280 POKESI+2+I*7,4000AND255:POKESI+3+I*7,4000/256:NEXT
290 POKESI+24,15+16+64        :POKESI+23,7:REM FILTER ENTFERNEN
300 GETA$:IFA$=""THEN300
310 FR=F(K(ASC(A$)))/M:FL=SI+V*7:W=FL+4:IFFR=ZTHEN420
320 POKEFL+6,Z
330 POKEFL+5,Z:REM HUELLKURVE AUSSCHALTEN
340 POKEW,8:POKEW,0:REM RESET
350 POKEFL,FR-HB*INT(FR/HB):REM FREQ LO
360 POKEFL+1,FR/HB:REM FREQ HI
370 POKEFL+6,HH:REM HALTEN
380 POKEW,WF+1:FORI=1TO50*AN:NEXT
390 POKEW,WF:REM AUS
400 IFP=1THENV=V+1:IFV=3THENV=0
410 GOTO300
415 REM FUNKTIONSTASTENABFRAGE
420 IFA$="{F1}"THENM=1:OK=4:GOTO300
430 IFA$="{F3}"THENM=2:OK=3:GOTO300
440 IFA$="{F5}"THENM=4:OK=2:GOTO300
450 IFA$="{F7}"THENM=8:OK=1:GOTO300
460 IFA$="{F2}"THENWF=16:GOTO300
470 IFA$="{F4}"THENWF=32:GOTO300
480 IFA$="{F6}"THENWF=64:GOTO300
490 IFA$="{F8}"THENWF=128:GOTO300
500 IFA$=" "THENP=1-P:HH=(HHANDNOT2)OR(NOTHHAND2):GOTO300
510 IFA$="{CLEAR}"THEN210
520 GOTO300
```

ANHANG K

ÜBERTRAGUNG VON FREMDEN BASIC-PROGRAMMEN AUF COMMODORE 64 BASIC

Besitzen Sie Programme, die in einer anderen BASIC-Version als COMMODORE BASIC geschrieben wurden, werden einige kleinere Anpassungen nötig sein, bevor sie auf dem COMMODORE 64 laufen können. Wir geben Ihnen nun einige Tips, die die Anpassung leichter machen.

Dimensionen von Strings

Entfernen Sie alle Statements, die die Länge eines Strings festlegen. Ein Befehl wie etwa DIM A$(I,J), der ein Stringarray für J Elemente der Länge I dimensioniert, muß in das COMMODORE BASIC Statement DIM A$(J) abgeändert werden.
Einige BASIC-Versionen benutzen ein Komma (,) oder Kaufmannsund (&) zur Verknüpfung von Strings. Diese müssen in ein Plus-Zeichen (+) geändert werden, das in COMMODORE BASIC der entsprechende Operator zur Stringverknüpfung ist.
Im BASIC des COMMODORE 64 dienen die Funktionen MID$, RIGHT$ und LEFT$ der Erzeugung von Teilstrings. Formen wie A$(I) zur Ansprache des I-ten Zeichens in String A$ oder A$(I,J) zur Gewinnung des Teilstrings von A$ von Position I bis J müssen wie folgt geändert werden:

sonstiges BASIC	**Commodore 64 BASIC**
`A$(l) = X$`	`A$ = LEFT$(A$,I-1)+X$+MID$(A$,I+1)`
`A$(I,J) = X$`	`A$ = LEFT$(A$,I-1)+X$+MID$(A$,J+1)`

Mehrfache Zuweisungen

Um die Variablen B und C gleichzeitig auf Null zu setzen, erlauben einige BASIC-Versionen Statements der Form:

```
10 LET B=C=0
```

COMMODORE 64 BASIC würde das zweite Gleichheitszeichen als logischen Operator interpretieren. Falls dann C=0 wäre, würde B= – 1. Schreiben Sie statt dessen zwei Befehle:

```
10 B=0 : C=0
```

Mehrfache Anweisungen

Einige BASIC-Versionen benutzen den Schrägstrich rückwärts (\) um mehrere Statements in einer Zeile voneinander zu trennen. In COMMODORE 64 BASIC werden alle Anweisungen durch einen Doppelpunkt (:) voneinander getrennt.

MAT-Funktionen

Programme, die die in einigen BASIC-Versionen vorrätigen MAT Funktionen für Matrizenoperationen verwenden, müssen umgeschrieben werden, indem diese Funktionen mit Hilfe von FOR ... NEXT Schleifen nachgebildet werden.

FEHLERMELDUNGEN

Dieser Anhang enthält eine vollständige Liste der Fehlermeldungen des COMMODORE 64 zusammen mit einer Beschreibung der Ursachen.

BAD DATA Von einem File wurden String Daten gelesen, das Programm erwartete jedoch numerische Daten.
BAD SUBSCRIPT Das Programm versuchte, ein Element eines Arrays anzusprechen, dessen Nummer außerhalb des in der DIM Anweisung vorgegebenen Bereichs liegt.
CAN'T CONTINUE Der Befehl CONT arbeitet nicht, wenn ein Programm nicht vorher mit RUN gestartet war, ein Fehler auftrat oder eine Zeile geändert wurde.
DEVICE NOT PRESENT Das angesprochene E/A Gerät war nicht verfügbar bei OPEN, CLOSE, CMD, PRINT#, INPUT# oder GET#.
DIVISION BY ZERO Division durch Null ist mathematisch undefiniert und nicht erlaubt.
EXTRA IGNORED Nach Aufforderung durch INPUT wurden zuviele Daten eingegeben. Nur die ersten wurden berücksichtigt.
FILE NOT FOUND Suchen Sie ein File auf Band, dann wurde eine END-OF-TAPE Markierung gefunden. Suchen Sie ein File auf der Diskette, dann existiert ein File dieses Namens nicht.
FILE NOT OPEN das mit CMD, PRINT#, INPUT#, GET# angesprochene File muß zuerst mit OPEN geöffnet werden.
FILE OPEN Sie versuchten ein File zu öffnen und benutzten dazu eine logische Filenummer, die bereits vergeben war.
FORMULA TOO COMPLEX Der Stringausdruck sollte in wenigstens zwei Teile aufgespalten werden, damit das System ihn bearbeiten kann.
ILLEGAL DIRECT INPUT kann nur innerhalb eines Programms benutzt werden und nicht im Direktmodus.
ILLEGAL QUANTITY Eine Zahl, die als Argument einer Funktion oder einer Anweisung benutzt wurde, liegt außerhalb des erlaubten Bereichs.
LOAD Es gibt ein Problem mit dem Programm auf der Kassette.
NEXT WITHOUT FOR Entweder wurden einige Schleifen nicht korrekt verschachtelt oder eine bei NEXT angegebene Variable entspricht nicht der bei FOR verwendeten.

NOT INPUT FILE Es wurde versucht mit INPUT# oder GET# Daten von einem File zu lesen, das nur zur Ausgabe bestimmt ist.

NOT OUTPUT FILE Sie versuchten Daten durch PRINT# an ein File zu senden, das nur zum Lesen geöffnet wurde.

OUT OF DATA Eine READ Anweisung wurde ausgeführt, es gibt aber keine Daten in einer DATA Zeile, die noch nicht mit READ gelesen wurden.

OUT OF MEMORY Es ist kein RAM Bereich mehr für Programm oder Variablen verfügbar. Dieser Fehler kann auch auftreten, wenn zu viele FOR ... NEXT Schleifen oder Unterprogramme ineinander geschachtelt oder zu viele Klammern geöffnet wurden.

OVERFLOW Das Ergebnis einer Rechnung ist größer als die größte erlaubte Zahl (1.70141183 E + 38).

REDIM'D ARRAY Ein Array kann nur einmal DIMensioniert werden. Wird eine Array Variable aufgerufen, bevor sie DIMensioniert wurde, führt der Rechner eine automatische DIM Operation aus, wobei die Dimension auf zehn gesetzt wird. Jede folgende DIM Anweisung wird dann diesen Fehler verursachen.

REDO FROM START String Zeichen wurden eingegeben, während ein INPUT Statement numerische Eingabe erwartete. Tippen Sie einfach die korrekten Eingaben noch einmal und das Programm wird von selbst fortfahren.

RETURN WITHOUT GOSUB Eine RETURN Anweisung wurde entdeckt aber kein GOSUB Befehl wurde vorher gegeben.

STRING TOO LONG Ein String kann höchstens 255 Zeichen enthalten.

SYNTAX Eine Anweisung kann vom Commodore 64 nicht erkannt werden. Sie haben eine Klammer vergessen oder zuviel angegeben, ein Schlüsselwort falsch eingetippt, usw.

TYPE MISMATCH Dieser Fehler tritt auf, wenn Sie eine Zahl statt eines Strings verwenden und umgekehrt.

UNDEF'D FUNKTION Sie nehmen Bezug auf eine selbst definierte Funktion, die noch nicht im DEF FN angelegt wurde oder deren Definitions-Zeile vom Programm noch nicht durchlaufen wurde.

UNDEF'D STATEMENT Eine nicht existente Zeilennummer wurde mit GOTO, GOSUB oder RUN angesprochen.

VERIFY Das Programm auf Band oder Diskette stimmt nicht mit dem Programm im Speicher überein.

ANHANG M

BIBLIOGRAPHIE

Christiani Kompakt-Kurs BASIC
P. Christiani (Hrsg.)
Technisches Lehrinstitut Dr.-Ing. P. Christiani, Konstanz

BASIC – Eine Einführung in das Programmieren
Rüdeger Baumann
Kiett, Stuttgart

Programmieren mit BASIC
Byron S. Gottfried
Mc. Graw Hill, Düsseldorf

Lerne BASIC mit dem Volkscomputer VC-20
Günter O. Hamann
Deutscher Betriebswirte-Verlag GmbH

BASIC in 100 Beispielen
Klaus Menzel
Teubner, Stuttgart

MOS-Programmierhandbuch zum 6502
MOS-Technology
Commodore Büromaschinen GmbH

CBM-Computer Handbuch
Osborn-Donahue
te-wi Verlag GmbH, München

6502, Programmieren in Assembler
Lance A. Leventhal
te-wi Verlag GmbH, München

Programmierung des 6502
Rodney Zaks
Sybex-Verlag GmbH, Düsseldorf

ANHANG N

SPRITE REGISTER ZUORDNUNG

Basisadresse VIC = 53248_{Dez} = $D000_{Hex}$

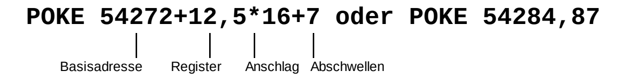
Register # Dez	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7	S0X6	S0X5	S0X4	S0X3	S0X2	S0X1	S0X0	SPRITE 0 X
1	1	S0Y7							S0Y0	SPRITE 0 Y
2	2	S1X7							S1X0	SPRITE 1 X
3	3	S1Y7							S1Y0	SPRITE 1 Y
4	4	S2X7							S2X0	SPRITE 2 X
5	5	S2Y7							S2Y0	SPRITE 2 Y
6	6	S3X7							S3X0	SPRITE 3 X
7	7	S3Y7							S3Y0	SPRITE 3 Y
8	8	S4X7							S4X0	SPRITE 4 X
9	9	S4Y7							S4Y0	SPRITE 4 Y
10	A	S5X7							S5X0	SPRITE 5 X
11	B	S5Y7							S5Y0	SPRITE 5 Y
12	C	S6X7							S6X0	SPRITE 6 X
13	D	S6Y7							S6Y0	SPRITE 6 Y
14	E	S7X7							S7X0	SPRITE 7 X
15	F	S7Y7							S7Y0	SPRITE 7 Y
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Höchste Bits der X-Werte
17	11	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	RASTER
19	13	LPX7							LPX0	LIGHT PEN X
20	14	LPY7							LPY0	LIGHT PEN Y
21	15	SE7							SE0	SPRITE ENABLE (EIN/AUS)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	
23	17	SEXY7							SEXY0	SPRITE EXPAND Y

Register # Dez	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
24	18	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Speicherbereich für Bildschirm-zeichen
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Interrupt Request's
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Interrupt Request MASKS
27	1B	BSP7							BSP0	Hintergrund Sprite PRIORITÄT
28	1C	SCM7							SCM0	MULTICOLOR SPRITE SELECT
29	1D	SEXX 7							SEXX 0	SPRITE EXPAND X
30	1E	SSC7							SSC0	Sprite-Sprite KOLLISION
31	1F	SBC7							SBC0	Sprite-Hintergrund KOLLISION
		FARBINFORMATION								
32	20									Bildrand
33	21									Hintergrund 0
34	22									Hintergrund 1
35	23									Hintergrund 2
36	24									Hintergrund 3
37	25									SMC0 (Sprite Multicolor)
38	26									SMC1 (Sprite Multicolor)
39	27									Farbe Sprite 0
40	28									Farbe Sprite 1
41	29									Farbe Sprite 2
42	2A									Farbe Sprite 3
43	2B									Farbe Sprite 4
44	2C									Farbe Sprite 5
45	2D									Farbe Sprite 6
46	2E									Farbe Sprite 7

Die Farbinformation entnehmen Sie bitte der Tabelle auf Seite 139. Im Multicolor-Modus dürfen nur die Farbcodes 0 ... 7 benutzt werden.

COMMODORE 64 KLANG KONTROLLEN

Diese Tabelle gibt Ihnen die Schlüsselzahlen, die Sie in Ihren Geräuschprogrammen benötigen, basierend auf den drei Stimmen des COMMODORE 64. Um eine Klangkontrolle von BASIC aus zu setzen, benötigen Sie Befehle von der Form POKE (Register), (Inhalt) (vgl. Beispiel unten).
Bei den geteilten Registern müssen Sie alle ausgewählten Werte addieren; z. B. für mittleren Anstieg, mittleres Abschwellen in Stimme 2:



Beachten Sie, daß Sie die Lautstärke (VOLUME) setzen müssen, bevor Sie einen Ton erzeugen können. POKE 54296 gefolgt von einer Zahl zwischen 0 und 15 setzt die Lautstärke für alle 3 Stimmen.

KONTROLLREGISTER FÜR TONERZEUGUNG

Basisadresse des SID: 54272_{Dez} = $D400_{Hex}$

<table>
<tr><th></th><th colspan="3">REGISTER</th><th colspan="8">INHALT</th></tr>
<tr><td>STIMME</td><td>1</td><td>2</td><td>3</td><td colspan="8"></td></tr>
<tr><td></td><td>0</td><td>7</td><td>14</td><td colspan="8">FREQUENZ, LO-BYTE (0 ... 255)</td></tr>
<tr><td></td><td>1</td><td>8</td><td>15</td><td colspan="8">FREQUENZ, HI-BYTE (0 ... 255)</td></tr>
<tr><td></td><td>2</td><td>9</td><td>16</td><td colspan="8">TASTVERHÄLTNIS, LO-BYTE (0 ... 255) (NUR FÜR RECHTECK)</td></tr>
<tr><td></td><td>3</td><td>10</td><td>17</td><td colspan="8">TASTVERHÄLTNIS, HI-BYTE (0 ... 15) (NUR FÜR RECHTECK)</td></tr>
<tr><td></td><td>4</td><td>11</td><td>18</td><td colspan="2">Wellenform:</td><td colspan="2">RAUSCHEN
129</td><td>RECHTECK
65</td><td colspan="2">SÄGEZAHN
33</td><td>DREIECK
17</td></tr>
<tr><td></td><td>5</td><td>12</td><td>19</td><td colspan="4">ANSCHLAG
0*16 (hart) ... 15*16 (weich)</td><td colspan="4">ABSCHWELLEN
0 (hart) ... 15*16 (weich)</td></tr>
<tr><td></td><td>6</td><td>13</td><td>20</td><td colspan="4">HALTEN
0*16 (stumm) ... 15*16 (laut)</td><td colspan="4">AUSKLINGEN
0 (schnell) ... 15 (langsam)</td></tr>
<tr><td></td><td>24</td><td>24</td><td>24</td><td colspan="8">LAUTSTÄRKE: 0 (stumm) ... 15 (volle Lautstärke)</td></tr>
</table>

Beispiel: Dauerton C-5 auf Stimme 2, Wellenform Dreieck:

```
SI = 54272
POKE SI+24,15:POKE SI+7,207:POKE SI+8,34: POKE SI+13,240
```
(Lautstärke) (Frequenz Lo) (Frequenz Hi) (laut halten, 15*16)

Einschalten des Tones: **POKE SI+11,17**
Ausschalten: **POKE SI+11,0**

Weitere Register des SID

REGISTER	INHALT							
21	GRENZFREQUENZ FILTER, LO-BYTE (0 ... 7)							
22	GRENZFREQUENZ FILTER, HI-BYTE (0 ... 255)							
23	RESONANZ 0 (keine) ... 15*16 (stark)				FILTER EINSCHALTEN extern 8	Sti 3 4	Sti 2 2	Sti 1 1
24	FILTER-MODUS Sti 3 aus 128	Hoch Paß 64	Band Paß 32	Tief Paß 16	LAUTSTÄRKE 0 (stumm) ... 15 (laut)			

Zusätzlich besitzt der SID noch 4 Register, die nichts mit der Tonerzeugung zu tun haben, sondern aus denen der Mikroprozessor des COMMODORE 64 bestimmte Informationen lesen kann:

REGISTER	INHALT
25	PADDLE X
26	PADDLE Y
27	OSZILLATOR 3
28	HÜLLKURVE 3

Um z.B. die Stellung der am Steuereingang angeschlossenen Paddle abzulesen, benutzen Sie die Befehle:

```
SI=54272:X=PEEK(SI+25):Y=PEEK(SI+26)
```

Die Variablen X und Y enthalten dann Werte zwischen 0 und 255, abhängig von der Stellung der Paddle-Drehknöpfe.
In den Registern 27 und 28 kann der Momentanwert des Oszillators und des Hüllkurvengenerators von Stimme 3 gelesen werden, z. B. um Zufallsgeneratoren zu erzeugen oder um die anderen Stimmen mit diesen Werten zu beeinflussen, um besondere Klangeffekte zu erzielen.

Mit Hilfe dieser Einstellungen können Sie verschiedene Instrumente nachahmen:

Instrument	Wellenform		Anschlag	Halten	Tastverhältnis
Piano	Puls	65	9	0	HI-0, LO-255
Flöte	Dreieck	17	96	0	
Cembalo	Sägezahn	33	9	0	
Xylophon	Dreieck	17	9	0	
Orgel	Dreieck	17	0	240	
Akkordeon	Dreieck	17	102	0	
Trompete	Sägezahn	33	96	0	

BEMERKUNG: Die Einstellungen für die Hüllkurve sollten immer gePOKEt werden **bevor** die Wellenform gePOKEt wird.

ANHANG P

WERTE FÜR MUSIK-NOTEN

In diesem Anhang finden Sie eine vollständige Liste der Noten und Werte, die in die Register FREQ HI und FREQ LO des Klangchips gePOKEt werden müssen, um den gewünschten Ton zu erzeugen.

NOTE	OKTAVE	DEZIMAL	HI	LOW
0	C – 0	278	1	22
1	C# – 0	295	1	39
2	D – 0	313	1	57
3	D# – 0	331	1	75
4	E – 0	351	1	95
5	F – 0	372	1	116
6	F# – 0	394	1	138
7	G – 0	417	1	161
8	G#-0	442	1	186
9	A – 0	468	1	212
10	A# – 0	496	1	240
11	H – 0	526	2	14
16	C – 1	557	2	45
17	C# – 1	590	2	78
18	D – 1	625	2	113
19	D# – 1	662	2	150
20	E – 1	702	2	190
21	F – 1	743	2	231
22	F# – 1	788	3	20
23	G – 1	834	3	66
24	G#-1	884	3	116
25	A – 1	937	3	169
26	A# – 1	992	3	224
27	H – 1	1051	4	27
32	C – 2	1114	4	90
33	C# – 2	1180	4	156
34	D – 2	1250	4	226
35	D# – 2	1325	5	45
36	E – 2	1403	5	123
37	F – 2	1487	5	207
38	F# – 2	1575	6	39
39	G – 2	1669	6	133
40	G# – 2	1768	6	232
41	A – 2	1873	7	81
42	A# – 2	1985	7	193
43	H – 2	2103	8	55
48	C – 3	2228	8	180
49	C# – 3	2360	9	56
50	D – 3	2500	9	196
51	D# – 3	2649	10	89
52	E – 3	2807	10	247
53	F – 3	2974	11	158
54	F# – 3	3150	12	78

NOTE	OKTAVE	DEZIMAL	HI	LOW
55	G – 3	3338	13	10
56	G# – 3	3536	13	208
57	A – 3	3746	14	162
58	A# – 3	3969	15	129
59	H – 3	4205	16	109
64	C – 4	4455	17	103
65	C# – 4	4720	18	112
66	D – 4	5001	19	137
67	D# – 4	5298	20	178
68	E – 4	5613	21	237
69	F – 4	5947	23	59
70	F# – 4	6301	24	157
71	G – 4	6676	26	20
72	G# – 4	7072	27	160
73	A – 4	7493	29	69
74	A# – 4	7939	31	3
75	H – 4	8411	32	219
80	C – 5	8911	34	207
81	C# – 5	9441	36	225
82	D – 5	10002	39	18
83	D# – 5	10597	41	101
84	E – 5	11227	43	219
85	F – 5	11894	46	118
86	F# – 5	12602	49	58
87	G – 5	13351	52	39
88	G# – 5	14145	55	65
89	A – 5	14986	58	138
90	A# – 5	15877	62	5
91	H – 5	16821	65	181
96	C – 6	17821	69	157
97	C# - 6	18881	73	193
98	D – 6	20004	78	36
99	D# – 6	21193	82	201
100	E – 6	22454	87	182
101	F – 6	23789	92	237
102	F# – 6	25203	98	115
103	G – 6	26702	104	78
104	G#-6	28290	110	130
105	A – 6	29972	117	20
106	A# – 6	31754	124	10
107	H – 6	33642	131	106
112	C – 7	35643	139	59
113	C# – 7	37762	147	130
114	D – 7	40008	156	72
115	D# – 7	42387	165	147
116	E – 7	44907	175	107
117	F – 7	47578	185	218
118	F# – 7	50407	196	231
119	G – 7	53404	208	156
120	G# – 7	56580	221	4
121	A – 7	59944	234	40
122	A# – 7	63508	248	20

Sie sind nicht an die Werte dieser Tabelle gebunden! Wenn Sie mehrere Stimmen benutzen, sollten Sie sogar bewußt die zweite und dritte Stimme etwas "verstimmen", d. h. das Lo-Byte aus der Tabelle leicht (!) abändern. Sie bekommen so einen volleren Klang.

ANHANG Q

Speicherbelegung des Commodore 64

(* =nützliche Adressen)

	Hex	Dezimal	Beschreibung
	0000	0	6510 Daten Richtungsregister
	0001	1	6510 Ausgaberegister
	0002	2	nicht benutzt
	0003 – 0004	3 – 4	Vektor zur Umrechnung Float – Fixed
	0005 – 0006	5 – 6	Vektor zur Umrechnung Fixed – Float
	0007	7	Suchzeichen
	0008	8	Flag für Gänsefüßchen Modus
	0009	9	TAB Spaltenzähler
	000A	10	0 = LOAD, 1 = VERIFY
	0008	11	Zeiger für Eingabepuffer / Anzahl Elemente
	000C	12	Flag für Standard DIM
	000D	13	Typ: FF = String, 00 = numerisch
	000E	14	Typ: 80 = Integer, 00 = Fließkomma
	000F	15	Flag für DATA / LIST
	0010	16	Element / FNx Flag
	0011	17	00 = INPUT, 40 = GET, 98 = READ
	0012	18	Vorzeichen des ATN Flag für Gleichheit bei Vergleich
	0013	19	aktuelles E/A Gerät
*	0014 – 0015	20 – 21	Integer Wert
	0016	22	Zeiger auf temporären Stringstapel
	0017 – 0018	23 – 24	Letzter temporärer String Vektor
	0019 – 0021	25 – 33	Stapel für temporäre Strings
	0022 – 0025	34 – 37	Bereich für Hilfszeiger
	0026 – 002A	38 – 42	Bereich für Produkt bei Multiplikation
*	002B – 002C	43 – 44	Zeiger auf Basic Anfang
*	002D – 002E	45 – 46	Zeiger auf Variablen Anfang
*	002F – 0030	47 – 48	Zeiger auf Beginn der Arrays
*	0031 – 0032	49 – 50	Zeiger auf Ende der Arrays
*	0033 – 0034	51 – 52	Zeiger auf Stringspeicher (bewegt sich abwärts)
	0035 – 0036	53 – 54	Hilfszeiger für Strings
*	0037 – 0038	55 – 56	Zeiger auf Speichergrenze
	0039 – 003A	57 – 58	Nummer der aktuellen Basic Zeile

	003B – 003C	59 – 60	Nummer der vorhergehenden Basic Zeile
	003D – 003E	61 – 62	Zeiger auf Basic Statement für CONT
	003F – 0040	63 – 64	Nummer der aktuellen DATA Zeile
	0041 – 0042	65 – 66	Adresse des aktuellen DATA Elements
*	0043 – 0044	67 – 68	Sprungvektor für Input
	0045 – 0046	69 – 70	aktueller Variablenname
	0047 – 0048	71 – 72	Adresse der aktuellen Variablen
	0049 – 004A	73 – 74	Variablenzeiger für FOR... NEXT
	004B – 004C	75 – 76	Zwischenspeicher für Basic Zeiger
	004D	77	Akkumulator für Vergleichssymoble
	004E – 0053	78 – 83	verschieden genutzter Arbeitsbereich (Zeiger, usw.)
	0054 – 0056	84 – 86	Sprungvektor für Funktionen
	0057 – 0060	87 – 96	verschieden genutzter Bereich für numerische Operationen
*	0061	97	Fließkomma Akkumulator #1 (FAC): Exponent
*	0062 – 0065	98 – 101	Fließkomma Akkumulator #1 (FAC): Mantisse
*	0066	102	Fließkomma Akkumulator #1 (FAC): Vorzeichen
	0067	103	Zeiger für Polynom Auswertung
	0069 – 006E	105 – 110	Fließkomma Akkumulator#2 Exponent usw.
	006F	111	Vorzeichen Vergleich Akku #1 / Akku #2
	0070	112	niederwertige Stelle Akku #1 (Rundung)
	0071 – 0072	113 – 114	Länge des Kassettenpuffers
*	0073 – 008A	115 – 138	CHRGET Subroutine (hole ein Zeichen)
	007A – 007B	122 – 123	Basic Zeiger innerhalb der Subroutine
	008B – 008F	139 – 143	Startwert des Zufallgenerators
*	0090	144	Statusbyte ST
	0091	145	Flag für STOP und RVS Taste
	0092	146	Zeitkonstante für Kassette
	0093	147	0 = LOAD, 1 = VERIFY
	0094	148	serieller Ausgang: Flag für zurückgestelltes Zeichen
	0095	149	zurückgestelltes Zeichen

	0096	150	EOT von Kassette empfangen (Cassette Sync #)
	0097	151	Speicher für Register
*	0098	152	Anzahl offener Files
*	0099	153	Eingabegerät (normal = 0)
*	009A	154	Ausgabe (CMD) Gerät (normal = 3)
	009B	155	Paritätsbyte von Band
	009C	156	Flag für Byte empfangen
	009D	157	Ausgabe Kontrolle (80 = direkt, 00 = RUN)
	009E	158	Fehler vom Band / Zeichenpuffer
	009F	159	Fehler vom Band korrigiert
*	00A0 – 00A2	160 – 162	interne Uhr (HML)
	00A3	163	serieller Bit Zähler Flag für EOI
	00A4	164	Zyklen Zähler
	00A5	165	Abwärtszähler beim Schreiben auf Kassette
	00A6	166	Zeiger auf Kassettenpuffer
	00A7 – 00AB	167 – 171	Flags für Schreiben und Lesen von Kassette
	00AC – 00AD	172 – 173	Zeiger für Prg.Start
	00AE – 00AF	174 – 175	Zeiger für Programmende
	00B0 – 00B1	176 – 177	Zeitkonstanten für Band
*	00B2 – 00B3	178 – 179	Zeiger auf Anfang des Kassettenpuffers
	00B4	180	Band Timer (1 = gesetzt), Bit Zähler
	00B5	181	Band EOT / RS 232 nächstes Bit zum Senden
	00B6	182	***
*	00B7	183	Anzahl Zeichen im Filenamen
*	00B8	184	aktuelle logische Filenummer
*	00B9	185	aktuelle Sekundäradresse
*	00BA	186	aktuelles Gerät
*	00BB – 00BC	187 – 188	Zeiger auf Filenamen
	00BD	189	***
	00BE	190	Anzahl zum Lesen/Schreiben verbleibender Blocks
	00BF	191	serieller Wortpuffer
	00C0	192	Kassettenmotor Flag
	00C1 – 00C2	193 – 194	E/A Startadresse

	00C3 – 00C4	195 – 196	Zeiger auf Vektoradressen des KERNAL
*	00C6	198	Anzahl Zeichen im Tastaturpuffer
*	00C7	199	RVS Flag für Bildschirm
	00C8	200	Zeiger auf Zeilenende für Eingabe
	00C9 – 00CA	201 – 202	Position des Eingabecursors (Zeile, Spalte)
*	00CB	203	gedrückte Taste (64=keine Taste)
	00CC	204	Cursor an/aus (0=Cursor blinkt)
	00CD	205	Zähler für blinkenden Cursor
	00CE	206	Zeichen in Cursorposition
	00CF	207	Cursor in Blinkphase
	00D0	208	Eingabe von Bildschirm/Tastatur
*	00D1 – 00D2	209 – 210	Zeiger auf Bildschirmzeile
*	00D3	211	Zeiger auf Bildschirmspalte
	00D4	212	0=direkter Cursor, sonst programmiert (QUOTE MODE)
*	00D5	213	Länge der aktuellen Bildschirmzeile (40/80)
*	00D6	214	Zeile, in der sich der Cursor befindet
	00D7	215	Letzte Taste/Prüfsumme/Puffer
*	00D8	216	Anzahl ausstehender Inserts
*	00D9 – 00F0	217 – 240	Bildschirmzeilen Verknüpfungstabelle
	00F1	241	unechte Bildschirmzeile
	00F2	242	Bildschirmzeilen Marke
*	00F3 – 00F4	242 – 244	Zeiger auf Bildschirm Farbe
	00F5 – 00F6	245 – 246	Zeiger auf Tastatur Decodiertabelle
	00F7 – 00F8	247 – 248	RS232 Empfangszeiger
	00F9 – 00FA	249 – 250	RS232 Übertragungszeiger
*	00FB – 00FE	251 – 254	Freier Platz in Page 0 für Betriebssystem
	00FF	255	Basic Speicher
	0100 – 010A	256 – 266	Arbeitsbereich zur Umwandlung von Fließkomma in ASCII
	0100 – 013E	256 – 318	Bandfehler
	0100 – 01FF	256 – 511	Bereich des Prozessor Stapels
*	0200 – 0258	512 – 600	Basic Eingabepuffer
*	0259 – 0262	601 – 610	Tabelle der logischen Files
*	0263 – 026C	611 – 620	Tabelle der Gerätenummern
*	026D – 0276	621 – 630	Tabelle der Sekundäradressen
*	0277 – 0280	631 – 640	Tastaturpuffer

*	0281 – 0282	641 – 642	Startadresse des RAM für Betriebssystem
*	0283 – 0284	642 – 644	Ende des RAM für Betriebssystem
	0285	645	Flag für Zeitüberschreitung auf seriellem Bus
*	0286	646	aktueller Farbcode
	0287	647	Farbe unter Cursor
*	0288	648	Bildschirmspeicher Adresse (Page)
*	0289	649	maximale Größe des Tastaturpuffers
*	028A	650	Tastenwiederholung (128=alle Tasten)
*	028B	651	Zähler für Wiederholungs-geschwindigkeit
	028C	652	Zähler für Wiederholungsverzögerung
*	028D	653	Flag für SHIFT/CNTRL
	028E	654	letztes SHIFT Muster der Tastatur
	028F – 0290	655 – 656	Zeiger auf Tastatur Decodiertabelle
*	0291	657	SHIFT Modus (0=gesetzt, 128=blockiert)
	0292	658	automat. Scrolling abwärts (0=ein ≠ 0 = aus)
	0293	659	RS232 Kontrollregister
	0294	660	RS232 Befehlsregister
	0285 – 0296	661 – 662	nicht Standard (Bit Zeit) ***
	0297	663	RS232 Statusregister
	0298	664	Anzahl zu sendender Bits
	0299 – 029A	665 – 666	Baud Rate
	029B	667	RS232 Empfangszeiger (receive)
	029C	668	RS232 Eingabezeiger (receive)
	029D	669	RS232 Übertragungszeiger (send)
	029E	670	RS232 Ausgabezeiger (send)
	029F – 02A0	271 – 672	enthält IRQ-Vektor während Kassettenbetrieb
	02A1 – 02FF	673 – 767	***
*	0300 – 0301	768 – 769	Vector für Fehlermeldungen
	0302 – 0303	770 – 771	Vector für Basic Warmstart
	0304 – 0305	772 – 773	Umwandlung von Schlüsselwörtern in Tokens
	0306 – 0307	774 – 775	Umwandlung der Tokens in Schlüsselwörter

0308 – 0309	776 – 777	neuen Basic Befehl ausführen
030A – 030B	778 – 779	arithmetisches Element holen
030C	780	Speicher für 6502 .A Register
030D	781	Speicher für 6502 .X Register
030E	782	Speicher für 6502 .Y Register
030F	783	Speicher für 6502 .P Register
0310 – 0313	784 – 787	USR Sprung
0314 – 0315	788 – 789	Hardware Interrupt (IRQ) (EA31)
0316 – 0317	790 – 791	Break Interrupt (FE66)
0318 – 0319	792 – 793	nicht maskierbarer Interrupt
		(NMI) (FE47)
031A – 031B	794 – 795	OPEN (F40A) (F34A)
031C – 031D	796 – 797	CLOSE (F291)
031E – 031F	798 – 799	Kanal für Eingabe (F2C7) (F209)
0320 – 0321	800 – 801	Kanal für Ausgabe (F250)
0322 – 0323	802 – 803	Wiederherstellung der E/A (Löschen
		aller offenen Kanäle (F333)
0324 – 0325	804 – 805	INPUT (F157)
0326 – 0327	806 – 807	OUTPUT (F1CA)
0328 – 0329	808 – 809	STOP-Taste prüfen (F770) (F6ED)
032A – 032B	810 – 811	GET (F13E)
032C – 032D	812 – 813	Close aller Kanäle (F32F)
032E – 032F	814 – 815	Benutzer-IRQ (FE66)
0330 – 0331	816 – 817	RAM laden (F4A5)
0332 – 0333	818 – 819	RAM speichern (F5ED)
0334 – 033B	820 – 827	***
033C – 03FB	828 – 1019	Kassettenpuffer
0400 – 07FF	1024 – 2047	1K Bildschirmspeicher
(0400 – 07E7	1024 – 2023	Video Matrix)
(07F8 – 07FF	2040 – 2047	Zeiger für Sprites)
0800 – 9FFF	2048 – 40959	Basic Benutzerspeicher
A000 – BFFF	40960 – 49151	8K Basic ROM
C000 – CFFF	49152 – 53247	4K RAM

BILDSCHIRM-STEUERZEICHEN

Bei der Eingabe einer in Anführungszeichen eingeschlossenen Zeichenkette (String) von der Tastatur befindet sich der Rechner im "Platzhalter-Modus", der jeweils nach dem Eintippen eines Anführungszeichens ein- und ausgeschaltet wird. In dieser speziellen Betriebsart werden die Bildschirm-Steuerbefehle von der Tastatur, wie z.B. (Cursor Home), nicht ausgeführt, sondern als Steuerzeichen in die Zeichenkette eingefügt und gelangen erst beim Abruf des Strings durch einen PRINT-Befehl zur Ausführung.
Auf dem Bildschirm werden diese Steuerzeichen durch spezielle revers dargestellte Platzhalter-Symbole sichtbar gemacht (siehe unten). Man findet diese Symbole häufig in Programm-Listings, die von einem Matrix-Drucker erstellt wurden.
Beim Eintippen eines solchen Programmes ist also jeweils die dem Platzhalter-Symbol entsprechende Steuertaste, z.B. die (Home)-Taste, zu betätigen.
Die folgende Tabelle gibt an, wie das Platzhalter-Symbol auf dem Bildschirm oder Drucker dargestellt wird. Hierbei gibt die erste Spalte die Symbole im "Grafik-Modus" (Großbuchstaben/Grafikzeichen) an und die zweite Spalte die Symbole im "Text-Modus" (Groß-/Kleinbuchstaben).
Nach den Symbolen wird die Taste bzw. Tastenkombination angegeben, durch die das Symbol hervorgerufen wird und dahinter wird die Steuerfunktion angegeben.

Groß/Grafik	Klein/Groß	Tastenkombination	Steuerfunktion
♥	S	SHIFT + CLR/HOME	BILDSCHIRM WIRD GELÖSCHT
S	s	CLR/HOME	CURSOR LINKS OBEN
]	]	CRSR links/rechts	CURSOR SPALTE NACH RECHTS
|	|	SHIFT + CRSR links/rechts	CURSOR SPALTE NACH LINKS
Q	q	CRSR oben/unten	CURSOR SPALTE NACH UNTEN
●	Q	SHIFT + CRSR oben/unten	CURSOR SPALTE NACH OBEN
▜	P	CTRL + 1	SCHWARZ
E	e	CTRL + 2	WEISS
£	£	CTRL + 3	ROT
◥	▧	CTRL + 4	TÜRKIS
▚	▚	CTRL + 5	VIOLETT
↑	↑	CTRL + 6	GRÜN
←	←	CTRL + 7	BLAU
π	▒	CTRL + 8	GELB

Groß/Grafik	Klein/Groß	Tastenkombination	Steuerfunktion
♠	A	C= + 1	ORANGE
╭	U	C= + 2	BRAUN
╳	V	C= + 3	HELLROT
○	W	C= + 4	GRAU 1
♣	X	C= + 5	GRAU 2
│	Y	C= + 6	HELLGRÜN
♦	Z	C= + 7	HELLBLAU
┼	┼	C= + 8	GRAU 3
R	r	CTRL + 9	INVERTIERTE DARSTELLUNG EIN
▁	R	CTRL + 0	INVERTIERTE DARSTELLUNG AUS

ANHANG S

HOCHAUFLÖSENDE GRAFIK AUF DEM C 64

Der Befehlssatz des C64 enthält keine Grafik-Befehle, der Rechner ist jedoch bereits für Grafik vorbereitet. Die maximale Auflösung von 200 x 320 Bildpunkten läßt sich dadurch erreichen, daß man einen bestimmten Speicherbereich mit dem Bitmuster belegt, das dem gewünschten Graphen entspricht und diesen Bereich auf dem Bildschirm abbildet (sogenanntes Bitmapping). Zur Darstellung der 64000 Bildpunkte (Pixels) benötigt man also 8 KB Speicherplatz.

Der erste Schritt zum Erzeugen einer Hi-Res-Grafik besteht darin, das Bit 3 in der Adresse 53272 zu setzen (Zeile 10 des Programms); dadurch wird bewirkt, daß der Speicherbereich ab 8192 für das "Bitmapping" benutzt wird. Um diesen Bereich auf den Bildschirm abzubilden, muß nun noch das Bit 5 in 53265 gesetzt werden (Zeile 20 des Programms), das den "Bit-map-mode" einschaltet.
Nun besteht noch das Problem, einen vorgegebenen Graphen (also eine Funktion o.ä.) in ein Bitmuster umzurechnen. Eine einfache Methode, angewandt auf eine Sinusfunktion, zeigt das folgende Programm. Soll eine andere Funktion f(x) gezeichnet werden, so muß diese in Zeile 60 in der Form

Y = INT (f(x))

eingesetzt werden. Dabei sind für x und y folgende Grenzen zu berücksichtigen:

$0 < x < 319$ und $0 < y < 199$

Bem.: Andere Farbkombinationen lassen sich erzielen, wenn man den gepokeden Wert in Zeile 25 ändert. Die unteren 4 Bits (unteres Nybble) bestimmen die Bildschirmfarbe, die oberen 4 Bits (oberes Nybble) die Farbe des dargestellten Graphen.

```
5 REM SINUSKURVE
10 POKE53272,PEEK(53272)OR8       :REM BEREICH FUER BITMAPPING AB 8192
20 POKE53265,PEEK(53265)OR32      :REM BITMAPMODE WIRD EINGESCHALTET
25 FORK=0TO999:POKE1024+K,14:NEXT :REM WAHL DER FARBKOMBINATION
27 FORI=0TO7999:POKE8192+I,0:NEXT :REM BILDSCHIRM WIRD GELOESCHT
50 FORX=0TO319
60 Y=INT(100+80*SIN(X*2*{PI}/160)):REM FUNKTION
70 FORN=0TO24
80 IFY>N*8-1ANDY<(N+1)*8THENBY=8192+N*320+8*INT(X/8)+Y-8*N:N=24:GOTO100
85 :                              :REM NUMMER DES BYTES WIRD BERECHNET
90 NEXTN
100 BI=8*(1+INT(X/8))-X-1         :REM BITMUSTER WIRD BERECHNET
110 IFPEEK(BY)<>0THENPOKEBY,PEEK(BY)OR2^BI
120 IFPEEK(BY)=0THENPOKEBY,2^BI
130 NEXTX
150 GOTO150
```

ANHANG T

PRINT FRE(0)

Die Variable FRE(0) wird als sog. Ganzzahl- oder Integervariable abgespeichert und kann deshalb keine Werte annehmen, die größer als 32767 sind. Übersteigt die Größe des freien Speichers diesen Wert, so erhält man negative Werte.

Die korrekte Anzahl freier Bytes erhält man, wenn man zu dem angezeigten Wert 65535 hinzuaddiert.

ANHANG U

BELEGUNG DER FUNKTIONSTASTEN

Die Funktionstasten (f1 bis f8) entsprechen den ASCII-Codes 133 bis 140 (vergl. Liste der ASCII-Codes). Die Abfrage des Tastaturpuffers kann in etwa in Form einer GET-Schleife geschehen, wie im folgenden Programm gezeigt wird:

```
1 REM FUNKTIONSTASTEN
10 GETA$:IFA$=""THEN 10
20 IFA$=CHR$(133)THEN 40
30 GOTO 10
40 PRINT"F1-TASTE WURDE GEDRUECKT"
50 GOTO 10
```

ANHANG V

LADEN EINES C64-PROGRAMMES IN EIN CBM-GERÄT (3, 4, 8000er)

Sie geben vor dem Laden folgende Befehlszeile ein:

```
POKE40,1:POKE41,8:POKE8*256,0:NEW
```

Nach dem Drücken der RETURN-Taste können Sie das Programm dann normal laden und starten.

Bem.: Das Laden von CBM-Programmen in den 64er ist ohne weiteres möglich, da der C64 die Ladeadresse ignoriert und jedes Programm an Anfang BASIC-Programmspeicher (Adresse 2048) legt. Dies ist natürlich beschränkt auf Programme in der BASIC 2.0-Version.

ABFRAGE DER DREHREGLER UND STEUERKNÜPPEL

Abfrage der Drehregler (Paddels):

Sind die Drehregler in Port 1 eingesteckt, so werden die Werte, die den Stellungen der Potentiometer entsprechen, in die Adressen 54297 und 54298 eingelesen.
Das Drücken der Feuerknöpfe wird in der Adresse 56321 registriert. Dort wird Bit 2 bzw. Bit 3 gelöscht.
Die Drehregler in Port 2 benutzen für die Abfrage der Feuerknöpfe die Adresse 56320. Die Potentiometerpositionen werden in denselben Adressen registriert wie bei Port 1.
Die Umschaltung auf Port 2 geschieht durch Setzen von Bit 7 in Adresse 56320. Da dieses Bit jedoch bei jedem Interrupt zurückgesetzt wird, ist die Abfrage der Paddles am Port 2 nur in Verbindung mit einem Maschinenprogramm möglich.

Abfrage der Steuerknüppel (Joysticks)

Die Abfrage der Joysticks geschieht in den Adressen 56321 (Port 1) und 56320 (Port 2). Den 4 Richtungen des Joysticks (oben, unten, links, rechts) entspricht das Löschen der Bits 0, 1, 2 und 3. Bei Betätigen der Feuerknöpfe wird Bit 4 im jeweiligen Register gelöscht.

STICHWORTVERZEICHNIS

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

BESCHEINIGUNG DES HERSTELLERS

Hiermit wird bestätigt, daß die Personal-Computer

COMMODORE C-64

in Übereinstimmung mit den Bestimmungen der

Amtsblattverfügung Nr. 1046/1984

funk-entstört ist.

Der Deutschen Bundespost wurde das Inverkehrbringen dieses Gerätes angezeigt und die Berechtigung zur Überprüfung der Serie auf Einhaltung der Bestimmungen eingeräumt.

COMMODORE BÜROMASCHINEN GMBH

CERTIFICATE OF THE MANUFACTURER

Herewith we certify that our devices Personal-Computer

COMMODORE C-64

correspond to the regulations

Amtsblattverfügung Nr. 1046/1984

is eliminated of radio interference.

The German Bundespost has been informed that this unit is on the market and has got the right to check on the mass production if the limits are kept.

COMMODORE BUSINESS MACHINES LIMITED

COMMODORE 64 BASIC, Kurz-Übersicht

EINFACHE VARIABLEN

Typ	*Name*	*Bereich*
Real	XY	±1.70141183E+38
		±2.93873588E–39
Integer	XY%	±32767
String	XY$	0 ... 255 Zeichen

X ist ein Buchstabe (von A-Z). Y ist ein Buchstabe oder eine Ziffer (0-9). Variablen-Namen dürfen auch länger sein. aber nur die ersten beiden Zeichen werden vom C64 erkannt.

FELDER

Typ	*Name*
Dimension 1	XY(5)
Dimension 2	XY(5,5)
Dimension 3	XY(5,5,5)

Felder mit bis zu 11 Elementen (Index 0 ... 10) können ohne weiteres definiert werden. Grö8ere Felder müssen vorher DIMensioniert werden:

DIM A(X,Y,Z)	setzt die Maximalwerte für die Indices fest, reserviert Speicher für (X+1) (Y+1) (Z+1) Elemente. erstes Element: A(0,0,0)

RECHEN-OPERATIONEN

=	gleich
-	Vorzeichen oder Subtraktion
↑	Exponentiation
*	Multiplikation
/	Division
+	Addition

LOGISCHE OPERATIONEN

=	gleich
<>	ungleich
<	kleiner als
>	größer als
<=	kleiner oder gleich
>=	größer oder gleich
NOT	logisch "nicht"
AND	logisch "und"
OR	logisch "oder" (nicht ausschließend)

SYSTEMKOMMANDOS

LOAD "NAME"	lädt ein Programm von der Kassette
SAVE "NAME"	speichert ein Programm auf die Kassette
LOAD "NAME",8	lädt ein Programm von der Diskette
LOAD "NAME",8,1	lädt ein Programm ohne zu verschieben (z. B. ein Assembler-Programm)
SAVE "NAME",8	speichert ein Programm auf die Diskette
VERIFY "NAME"	prüft, ob das Programm richtig geSAVEt wurde
RUN	führt das Programm aus
RUN XXXXX	führt das Programm ab Zeile XXXXX aus
STOP	stoppt das Programm mit Zeilenangabe
END	beendet das Programm
CONT	setzt das Programm fort, wo es angehalten wurde
POKE X,Y	schreibt Y an die Speicheradresse X
SYS XXXXX	springt in ein Maschinenprogramm, das bei XXXXX anfängt
WAIT X,Y,Z	Programm wartet, bis der Inhalt der Speicherzelle X, geORt mit Y und geANDet mit Z, von Null verschieden ist.

SYSTEMFUNKTIONEN

PEEK(X)	Inhalt der Speicherzelle X
USR(X)	Ergebnis eines Maschinen-Unterprogramms, das den Parameter X übernimmt.
RND(X)	Zufallszahl
FRE(X)	Zahl der freien Speicherzellen

EDIER- UND FORMATIERKOMMANDOS

LIST	listet das ganze Programm
LIST A-B	listet die Programmzeilen von A bis B
REM Kommentar	alles was hinter REMark steht, beeinflußt den Programmablauf nicht
CLR/HOME	bringt den Cursor in die linke obere Ecke
SHIFT CLR/HOME	löscht den Bildschirm
SHIFT INST/DEL	fügt vor dem Cursor eine Leerstelle ein
INST/DEL	löscht das Zeichen vor dem Cursor
CTRL	mit Zifferntaste: Farbwahl (auch nach PRINT)
CRSR Tasten	bewegen den Cursor in die angegebene Richtung
COMMODORE Taste	mit SHIFT: Umschalten Grafik/ Kleinbuchstaben mit Zifferntaste: zusätzliche Farben

FORMATIONSFUNKTIONEN

TAB(X)	Zwischenraum bis Spalte X
SPC(X)	X Zwischenräume
POS(X)	Position des nächsten freien Platzes in der Zeile

STRINGFUNKTIONEN

LEN(X$)	Anzahl der Zeichen im String X$
STR$(X)	Ziffernfolge, die die Zahl X darstellt
VAL(X$)	numerischer Wert von X$ (bis zum ersten nicht numerischen Zeichen)
CHR$(X)	Zeichen zum ASCII-Code X
ASC(X$)	ASCII-Code zum ersten Zeichen von X$
LEFT$(A$,X)	die ersten X Zeichen von A$
RIGHT$(A$,X)	die letzten X Zeichen von A$
MID$(A$,X, Y)	Y Zeichen von A$, beginnend mit dem Xten

EINGABE/AUSGABE BEFEHLE

INPUT A$ bzw. A	gibt ein "?" aus und wartet auf die Eingabe eines Strings oder eines numerischen Wertes
INPUT "ABC";A	gibt zusätzlich Text "ABC" aus
GET A$ bzw. A	liest Zeichen von der Tastatur ein
DATA A,"B",C	Folge von Werten, die durch READ gelesen werden können
READ A$ bzw. A	ordnet der Variablen A$ bzw. A den nächsten Wert in der DATA-Liste zu
RESTORE	setzt den DATA-Zeiger auf den Anfang der Liste
PRINT "A="; A	gibt "A=" und den Wert von A auf dem Bildschirm aus. ';' unterdrückt Zwischenräume, ',' tabuliert zum nächsten Zehnerfeld
OPEN 1,4	öffnet File 1 zum Drucker
INPUT#2, A$	liest Variable A$ vom File 2
GET#2, A$	liest ein Zeichen in Variable A$ vom File 2
PRINT#1, A	gibt A auf dem Drucker aus
CLOSE 1	schließt File 1

PROGRAMMABLAUF

GOTO X	verzweigt zu Zeile X
IF A=3 THEN 10	wenn die Bedingung war ist, wird ausgeführt was hinter THEN steht, sonst Sprung zur nächsten Zeile
FOR A=1 TO 10 STEP 2	führt alle Befehle zwischen FOR und dem zugehörigen NEXT aus, wobei A alle Werte von 1 bis 10 in Zweiersprüngen annimmt. Wenn kein Step angegeben ist, wird Step 1 angenommen
NEXT A	markiert das Ende der Schleife (A kann weggelassen werden)
GOSUB 2000	verzweigt in ein Unterprogramm, das bei Zeile 2000 beginnt
RETURN	Ende des Unterprogramms. Fortsetzung hinter dem letzten GOSU8
ON X GOTO A,B,C ON X GOSUB A,B,C	verzweigt zur Xten der Zeilen A,B,C. d, h. wenn X=1, GOTO A etc.



Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Langenhagstraße 1
CH-4147 Aesch

Commodore GmbH
Kinskygasse 40-44
A-1232 Wien

Artikel-Nr. 556400

Änderungen vorbehalten